

# **ADSP-BF60x Blackfin® Processor Hardware Reference**

Preliminary Revision 0.41, June 2012

Part Number  
82-100113-01

Analog Devices, Inc.  
One Technology Way  
Norwood, Mass. 02062-9106



## **Copyright Information**

© 2012 Analog Devices, Inc., ALL RIGHTS RESERVED. This document may not be reproduced in any form without prior, express written consent from Analog Devices, Inc.

Printed in the USA.

## **Disclaimer**

Analog Devices, Inc. reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent rights of Analog Devices, Inc.

## **Trademark and Service Mark Notice**

The Analog Devices logo, Blackfin, EZ-KIT Lite, SHARC, TigerSHARC, VisualDSP++, and CrossCore Embedded Studio are registered trademarks of Analog Devices, Inc.

All other brand and product names are trademarks or service marks of their respective owners.

# Contents

## Contents

### Preface

|                                                      |          |
|------------------------------------------------------|----------|
| Purpose of This Manual .....                         | -lxxix   |
| Intended Audience .....                              | -lxxix   |
| Manual Contents .....                                | -lxxix   |
| What's New in This Manual .....                      | -lxxxii  |
| Technical or Customer Support .....                  | -lxxxii  |
| Registration for MyAnalog.com .....                  | -lxxxiii |
| EngineerZone .....                                   | -lxxxiii |
| Social Networking Web Sites .....                    | -lxxxiii |
| Supported Processors .....                           | -lxxxiii |
| Product Information .....                            | -lxxxiv  |
| Analog Devices Web Site .....                        | -lxxxiv  |
| CrossCore Embedded Studio Online Documentation ..... | -lxxxiv  |
| Technical Library CD .....                           | -lxxxv   |
| Notation Conventions .....                           | -lxxxv   |
| Register Diagram Conventions .....                   | -lxxxvii |

### Introduction

|                                   |     |
|-----------------------------------|-----|
| Blackfin Processor Core .....     | 1-2 |
| Instruction Set Description ..... | 1-4 |
| Processor Safety Features .....   | 1-5 |
| Dual Core Supervision .....       | 1-5 |

|                                              |      |
|----------------------------------------------|------|
| Fault Management .....                       | 1-5  |
| System Protection .....                      | 1-6  |
| Bandwidth Monitor .....                      | 1-6  |
| Private (to each Core) Memory .....          | 1-6  |
| Shared (by both Cores) Memory .....          | 1-6  |
| I/O Memory Space .....                       | 1-7  |
| Memory Protection .....                      | 1-7  |
| Multi-Parity-Bit-Protected L1 Memories ..... | 1-7  |
| ECC-Protected L2 Memory .....                | 1-7  |
| CRC-Protected Memories .....                 | 1-7  |
| Watchpoint Protection .....                  | 1-8  |
| Pin Multiplexing .....                       | 1-8  |
| Processor Infrastructure .....               | 1-8  |
| System Crossbar (SCB) .....                  | 1-8  |
| Clock Generation .....                       | 1-8  |
| Crystal Oscillator (SYS_XTAL) .....          | 1-9  |
| Clock Out/External Clock .....               | 1-9  |
| System Protection Unit (SPU) .....           | 1-9  |
| Dynamic Power Management (DPM) .....         | 1-10 |
| Core Timers .....                            | 1-10 |
| Event Handling .....                         | 1-11 |
| Core Event Controller (CEC) .....            | 1-11 |
| System Event Controller (SEC) .....          | 1-11 |
| Trigger Routing Unit (TRU) .....             | 1-12 |
| Pin Interrupts .....                         | 1-12 |
| Memory Architecture .....                    | 1-12 |
| Static Memory Controller (SMC) .....         | 1-13 |
| L2 Memory Controller .....                   | 1-13 |
| Dynamic Memory Controller (DMC) .....        | 1-13 |
| Cyclic Redundancy Check (CRC) .....          | 1-14 |



|                                                          |      |
|----------------------------------------------------------|------|
| Direct Memory Access (DMA) .....                         | 1-14 |
| On Chip Peripherals .....                                | 1-15 |
| General-Purpose I/O (GPIO) .....                         | 1-16 |
| General-Purpose Timers .....                             | 1-16 |
| Watchdog Timers .....                                    | 1-16 |
| General-Purpose Counters .....                           | 1-17 |
| Pulsewidth Modulator (PWM) .....                         | 1-17 |
| Universal Asynchronous Receiver/Transmitter (UART) ..... | 1-18 |
| 2-Wire Interface (TWI) .....                             | 1-19 |
| Controller Area Network (CAN) .....                      | 1-19 |
| Universal Serial Bus (USB) .....                         | 1-20 |
| Ethernet Media Access Controller (MAC) .....             | 1-20 |
| Removable Storage Interface (RSI) .....                  | 1-22 |
| Serial Peripheral Interface (SPI) .....                  | 1-22 |
| Serial Port (SPORT) .....                                | 1-22 |
| ADC Control Module (ACM) .....                           | 1-23 |
| Link Port (LP) .....                                     | 1-24 |
| Video Sub-System and Pixel Pipeline (PxP) .....          | 1-24 |
| Pipelined Vision Processor (PVP) .....                   | 1-25 |
| Parallel Peripheral Interface (PPI) .....                | 1-26 |
| Pixel Compositor (PIXC) .....                            | 1-26 |
| Reset Control Unit (RCU) .....                           | 1-27 |
| Booting .....                                            | 1-28 |
| System Design .....                                      | 1-29 |
| System Debug Unit .....                                  | 1-29 |
| System Watchpoint Unit .....                             | 1-29 |

## System Crossbars (SCB)

|                                  |     |
|----------------------------------|-----|
| SCB Features .....               | 2-1 |
| SCB Functional Description ..... | 2-1 |
| SCB Register List .....          | 2-1 |

|                                                             |      |
|-------------------------------------------------------------|------|
| SCB Definitions .....                                       | 2-2  |
| SCB Block Diagram .....                                     | 2-2  |
| SCB Hierarchy Block Diagram .....                           | 2-3  |
| ADSP-BF60x Block Diagram .....                              | 2-3  |
| SCB Arbitration Tables .....                                | 2-4  |
| SCB Units, Master Interfaces, and Arbitration Types .....   | 2-5  |
| SCB0 Slots and Masters .....                                | 2-6  |
| SCB1 Slots and Masters .....                                | 2-14 |
| SCB2 Slots and Masters .....                                | 2-15 |
| SCB3 Slots and Masters .....                                | 2-16 |
| SCB4 Slots and Masters .....                                | 2-17 |
| SCB5 Slots and Masters .....                                | 2-17 |
| SCB6 Slots and Masters .....                                | 2-18 |
| SCB7 Slots and Masters .....                                | 2-18 |
| SCB8 Slots and Masters .....                                | 2-19 |
| SCB9 Slots and Masters .....                                | 2-19 |
| SCB Programming Model .....                                 | 2-20 |
| SCB Register Access Restrictions .....                      | 2-20 |
| SCB Programming Concepts .....                              | 2-20 |
| Arbitration Model Configuration .....                       | 2-20 |
| SCB Programming Examples .....                              | 2-21 |
| SCB Register Descriptions .....                             | 2-21 |
| Arbitration Read Channel Master Interface n Register .....  | 2-21 |
| Arbitration Write Channel Master Interface n Register ..... | 2-22 |
| Slave Interfaces Number Register .....                      | 2-23 |
| Master Interfaces Number Register .....                     | 2-24 |

## **Clock Generation Unit (CGU)**

|                                     |     |
|-------------------------------------|-----|
| CGU Functional Description .....    | 3-1 |
| CGU Register List .....             | 3-1 |
| ADSP-BF60x CGU Interrupt List ..... | 3-2 |

|                                                                                          |      |
|------------------------------------------------------------------------------------------|------|
| ADSP-BF60x CGU Trigger List .....                                                        | 3-2  |
| CGU Definitions .....                                                                    | 3-3  |
| CGU PLL Block Diagram .....                                                              | 3-3  |
| CGU Event Control .....                                                                  | 3-5  |
| CGU Event .....                                                                          | 3-5  |
| CGU Error .....                                                                          | 3-5  |
| CGU Generated Bus Errors .....                                                           | 3-6  |
| CGU Programming Model .....                                                              | 3-6  |
| CGU Mode Configuration .....                                                             | 3-6  |
| Changing the PLLCLK Frequency .....                                                      | 3-7  |
| Changing the CCLKn, SYSCLK, or SCLKn frequency Without Modifying the PLLCLK Frequency .. | 3-7  |
| Changing the DCLK Clock Frequency .....                                                  | 3-8  |
| Changing the OUTCLK Frequency .....                                                      | 3-9  |
| Aligning All Clocks .....                                                                | 3-9  |
| Valid ADSP-BF60x CGU_CTL.MSEL Bit Settings .....                                         | 3-10 |
| CGU Register Descriptions .....                                                          | 3-10 |
| Control Register .....                                                                   | 3-10 |
| Status Register .....                                                                    | 3-12 |
| Divisor Register .....                                                                   | 3-17 |
| CLKOUT Select Register .....                                                             | 3-20 |

## System Protection Unit (SPU)

|                                  |     |
|----------------------------------|-----|
| SPU Features .....               | 4-1 |
| SPU Functional Description ..... | 4-1 |
| SPU Register List .....          | 4-4 |
| SPU Definitions .....            | 4-4 |
| SPU Block Diagram .....          | 4-4 |
| SPU Architectural Concepts ..... | 4-5 |
| SPU Event Control .....          | 4-5 |
| SPU Programming Model .....      | 4-5 |
| SPU Mode Configuration .....     | 4-6 |

|                                       |      |
|---------------------------------------|------|
| Locking Write-Protect Registers ..... | 4-6  |
| Protecting a Peripheral .....         | 4-7  |
| SPU Register Descriptions .....       | 4-7  |
| Control Register .....                | 4-7  |
| Status Register .....                 | 4-8  |
| Write Protect Register n .....        | 4-10 |

## Dynamic Power Management (DPM)

|                                                                              |      |
|------------------------------------------------------------------------------|------|
| DPM Features .....                                                           | 5-1  |
| DPM Functional Description .....                                             | 5-1  |
| DPM Register List .....                                                      | 5-1  |
| ADSP-BF60x DPM Interrupt List .....                                          | 5-2  |
| DPM Definitions .....                                                        | 5-3  |
| DPM Operating Modes .....                                                    | 5-3  |
| RESET .....                                                                  | 5-4  |
| FULL-ON .....                                                                | 5-4  |
| ACTIVE .....                                                                 | 5-4  |
| ACTIVE with PLL Disabled .....                                               | 5-4  |
| DEEP SLEEP .....                                                             | 5-4  |
| HIBERNATE .....                                                              | 5-5  |
| Ensuring VDD_INT is Restored Before Booting .....                            | 5-6  |
| Using the Power Good Counter to Ensure That Internal Power is Restored ..... | 5-7  |
| Using SYS_PWRGD to Ensure That VDD_INT is Restored .....                     | 5-8  |
| DPM Event Control .....                                                      | 5-8  |
| DPM Events .....                                                             | 5-8  |
| DPM Errors .....                                                             | 5-8  |
| Configuring DPM DEEP SLEEP Mode .....                                        | 5-9  |
| Configuring DPM HIBERNATE Mode .....                                         | 5-9  |
| ADSP-BF60x Wake-Up Sources .....                                             | 5-10 |
| ADSP-BF60x Clock Buffer Disable Bit Assignments .....                        | 5-11 |
| ADSP-BF60x Hibernate Disable Bit Assignments .....                           | 5-11 |

|                                                |      |
|------------------------------------------------|------|
| DPM Register Descriptions .....                | 5-12 |
| Control Register .....                         | 5-13 |
| Status Register .....                          | 5-15 |
| Core Clock Buffer Disable Register .....       | 5-17 |
| Core Clock Buffer Enable Register .....        | 5-18 |
| Core Clock Buffer Status Register .....        | 5-19 |
| Core Clock Buffer Status Sticky Register ..... | 5-20 |
| System Clock Buffer Disable Register .....     | 5-21 |
| Wakeup Enable Register .....                   | 5-22 |
| Wakeup Polarity Register .....                 | 5-23 |
| Wakeup Status Register .....                   | 5-24 |
| Hibernate Disable Register .....               | 5-25 |
| Power Good Counter Register .....              | 5-26 |
| Restore n Register .....                       | 5-27 |

## Core Timers

|                                 |     |
|---------------------------------|-----|
| Core Timer Features .....       | 6-1 |
| TMR Register List .....         | 6-1 |
| Block Diagram .....             | 6-2 |
| External Interfaces .....       | 6-2 |
| Internal Interfaces .....       | 6-2 |
| Timer Operation .....           | 6-2 |
| Interrupt Processing .....      | 6-3 |
| TMR Register Descriptions ..... | 6-3 |
| Timer Control Register .....    | 6-4 |
| Timer Period Register .....     | 6-5 |
| Timer Scale Register .....      | 6-6 |
| Timer Count Register .....      | 6-6 |

## System Event Controller (SEC)

|                    |     |
|--------------------|-----|
| SEC Features ..... | 7-1 |
|--------------------|-----|

|                                                       |      |
|-------------------------------------------------------|------|
| SEC Functional Description .....                      | 7-1  |
| SEC Register List .....                               | 7-1  |
| ADSP-BF60x Interrupt List .....                       | 7-3  |
| SEC Definitions .....                                 | 7-11 |
| SEC Block Diagram .....                               | 7-11 |
| SFI Block Diagram .....                               | 7-12 |
| SCI Block Diagram .....                               | 7-13 |
| SSI Block Diagram .....                               | 7-13 |
| SEC Architectural Concepts .....                      | 7-14 |
| System Interrupt Acknowledge .....                    | 7-14 |
| System Interrupt Groups .....                         | 7-14 |
| System Interrupt Flow .....                           | 7-14 |
| SEC Error .....                                       | 7-16 |
| SEC Programming Model .....                           | 7-16 |
| Programming Concepts .....                            | 7-16 |
| Programming Examples .....                            | 7-16 |
| Configuring a System Source to Interrupt a Core ..... | 7-17 |
| Configuring a System Source as a Fault .....          | 7-17 |
| SEC Register Descriptions .....                       | 7-17 |
| SCI Control Register n .....                          | 7-18 |
| SCI Status Register n .....                           | 7-20 |
| Core Pending Register n .....                         | 7-22 |
| SCI Active Register n .....                           | 7-23 |
| SCI Priority Mask Register n .....                    | 7-24 |
| SCI Group Mask Register n .....                       | 7-25 |
| SCI Priority Level Register n .....                   | 7-26 |
| SCI Source ID Register n .....                        | 7-27 |
| Fault Control Register .....                          | 7-28 |
| Fault Status Register .....                           | 7-31 |
| Fault Source ID Register .....                        | 7-33 |

|                                                 |      |
|-------------------------------------------------|------|
| Fault End Register .....                        | 7-34 |
| Fault Delay Register .....                      | 7-35 |
| Fault Delay Current Register .....              | 7-36 |
| Fault System Reset Delay Register .....         | 7-37 |
| Fault System Reset Delay Current Register ..... | 7-37 |
| Fault COP Period Register .....                 | 7-38 |
| Fault COP Period Current Register .....         | 7-39 |
| Global Control Register .....                   | 7-39 |
| Global Status Register .....                    | 7-41 |
| Global Raise Register .....                     | 7-42 |
| Global End Register .....                       | 7-43 |
| Source Control Register n .....                 | 7-44 |
| Source Status Register n .....                  | 7-46 |

## Trigger Routing Unit (TRU)

|                                    |      |
|------------------------------------|------|
| TRU Features .....                 | 8-1  |
| TRU Functional Description .....   | 8-1  |
| TRU Register List .....            | 8-1  |
| ADSP-BF60x Trigger List .....      | 8-2  |
| TRU Definitions .....              | 8-11 |
| TRU Block Diagram .....            | 8-11 |
| TRU Architectural Concepts .....   | 8-12 |
| TRU Programming Model .....        | 8-12 |
| Programming Concepts .....         | 8-13 |
| Programming Examples .....         | 8-13 |
| TRU Event Control .....            | 8-13 |
| TRU Status and Error Signals ..... | 8-13 |
| TRU Register Descriptions .....    | 8-14 |
| Slave Select Register .....        | 8-14 |
| Master Trigger Register .....      | 8-15 |
| Error Address Register .....       | 8-16 |

|                                   |      |
|-----------------------------------|------|
| Status Information Register ..... | 8-17 |
| Revision ID Register .....        | 8-17 |
| Global Control Register .....     | 8-18 |

## Static Memory Controller (SMC)

|                                                                    |      |
|--------------------------------------------------------------------|------|
| SMC Functional Description .....                                   | 9-1  |
| Features .....                                                     | 9-1  |
| SMC Register List .....                                            | 9-1  |
| SMC Memory Controller Definitions .....                            | 9-2  |
| SMC Architectural Concepts .....                                   | 9-4  |
| Avoiding Bus Contention .....                                      | 9-4  |
| ARDY Input Control .....                                           | 9-5  |
| Bus Request and Bus Grant .....                                    | 9-6  |
| Bank-Off Bus Grant .....                                           | 9-6  |
| Bus Request and Bus Grant Protocol Timing .....                    | 9-7  |
| Disabling Bus Grant to External Memory Controllers .....           | 9-7  |
| SMC Operating Modes .....                                          | 9-8  |
| Asynchronous Flash Mode .....                                      | 9-8  |
| Synchronous Burst Mode .....                                       | 9-8  |
| Asynchronous Page Mode .....                                       | 9-9  |
| SMC Event Control .....                                            | 9-9  |
| SMC Errors .....                                                   | 9-9  |
| SMC Programmable Timing Characteristics .....                      | 9-9  |
| Asynchronous SRAM Reads and Writes .....                           | 9-9  |
| Asynchronous SRAM Reads with IDLE Transition Cycles Inserted ..... | 9-11 |
| High Speed Asynchronous SRAM Read Burst .....                      | 9-12 |
| High Speed Asynchronous SRAM Writes .....                          | 9-13 |
| Asynchronous SRAM Read with ARDY .....                             | 9-14 |
| Asynchronous Flash Read Operation .....                            | 9-16 |
| Asynchronous Flash Write Operation .....                           | 9-17 |
| Asynchronous Flash Page Mode Read operation .....                  | 9-18 |



|                                                  |      |
|--------------------------------------------------|------|
| Synchronous Burst Mode Read Operation .....      | 9-19 |
| Asynchronous FIFO Read and Write Operation ..... | 9-20 |
| SMC Programming Model .....                      | 9-22 |
| SMC Register Descriptions .....                  | 9-23 |
| Grant Control Register .....                     | 9-24 |
| Grant Status Register .....                      | 9-24 |
| Bank 0 Control Register .....                    | 9-26 |
| Bank 0 Timing Register .....                     | 9-29 |
| Bank 0 Extended Timing Register .....            | 9-32 |
| Bank 1 Control Register .....                    | 9-34 |
| Bank 1 Timing Register .....                     | 9-37 |
| Bank 1 Extended Timing Register .....            | 9-40 |
| Bank 2 Control Register .....                    | 9-42 |
| Bank 2 Timing Register .....                     | 9-45 |
| Bank 2 Extended Timing Register .....            | 9-48 |
| Bank 3 Control Register .....                    | 9-50 |
| Bank 3 Timing Register .....                     | 9-53 |
| Bank 3 Extended Timing Register .....            | 9-56 |

## **L2 Memory Controller (L2CTL)**

|                                                   |      |
|---------------------------------------------------|------|
| L2 Memory Controller Functional Description ..... | 10-1 |
| L2CTL Register List .....                         | 10-1 |
| ADSP-BF60x L2CTL Interrupt List .....             | 10-2 |
| L2 Memory Controller Block Diagram .....          | 10-3 |
| L2 Memory Controller Architectural Concepts ..... | 10-3 |
| Access Characteristics .....                      | 10-3 |
| Cache Support .....                               | 10-4 |
| L2 Read/Write Latency and Throughput .....        | 10-4 |
| Arbitration and Priority .....                    | 10-5 |
| L2 Memory Controller Event Control .....          | 10-6 |
| L2CTL Register Descriptions .....                 | 10-7 |

|                                                  |       |
|--------------------------------------------------|-------|
| Control Register .....                           | 10-8  |
| Access Control Core 0 Register .....             | 10-12 |
| Access Control Core 1 Register .....             | 10-14 |
| Access Control System Register .....             | 10-17 |
| Status Register .....                            | 10-19 |
| Read Priority Count Register .....               | 10-22 |
| Write Priority Count Register .....              | 10-23 |
| Refresh Address Register .....                   | 10-24 |
| ECC Error Address 0 Register .....               | 10-25 |
| ECC Error Address 1 Register .....               | 10-26 |
| ECC Error Address 2 Register .....               | 10-27 |
| ECC Error Address 3 Register .....               | 10-27 |
| ECC Error Address 4 Register .....               | 10-28 |
| ECC Error Address 5 Register .....               | 10-29 |
| ECC Error Address 6 Register .....               | 10-29 |
| ECC Error Address 7 Register .....               | 10-30 |
| Error Type 0 Register .....                      | 10-31 |
| Error Type 0 Address Register .....              | 10-33 |
| Error Type 1 Register .....                      | 10-33 |
| Error Type 1 Address Register .....              | 10-35 |
| BF60x-Specific Information .....                 | 10-35 |
| L2 Memory Controller ADSP-BF60x Throughput ..... | 10-36 |

## Dynamic Memory Controller (DMC)

|                                                     |      |
|-----------------------------------------------------|------|
| DMC Register List .....                             | 11-1 |
| ADSP-BF609 specific Register and bit settings ..... | 11-2 |
| DMC Features .....                                  | 11-2 |
| Feature Exclusions .....                            | 11-3 |
| Functional Description .....                        | 11-4 |
| DMC Protocol Controller .....                       | 11-4 |
| DMC Efficiency Controller .....                     | 11-5 |

|                                            |       |
|--------------------------------------------|-------|
| <b>Read/Write Turnaround</b>               | 11-5  |
| Closed Page Per Bank                       | 11-6  |
| SCB ID Based Priority                      | 11-6  |
| Delaying up to Eight Auto-Refresh Commands | 11-7  |
| Page Interleaving and Bank Interleaving    | 11-7  |
| System Crossbar Slave Interface            | 11-8  |
| Read/Write Command and Data Buffers        | 11-8  |
| Peripheral Bus Slave Interface             | 11-9  |
| Power Savings                              | 11-9  |
| DMC DMA                                    | 11-10 |
| DMC Programming Model                      | 11-11 |
| Configuring the DMC                        | 11-11 |
| DMC Register Descriptions                  | 11-14 |
| Control Register                           | 11-15 |
| Status Register                            | 11-19 |
| Efficiency Control Register                | 11-21 |
| Priority ID Register                       | 11-26 |
| Priority ID Mask Register                  | 11-27 |
| Configuration Register                     | 11-27 |
| Timing 0 Register                          | 11-29 |
| Timing 1 Register                          | 11-30 |
| Timing 2 Register                          | 11-32 |
| Mask (Mode Register Shadow) Register       | 11-33 |
| Shadow MR Register                         | 11-35 |
| Shadow EMR1 Register                       | 11-37 |
| Shadow EMR2 Register                       | 11-40 |
| Shadow EMR3 Register                       | 11-42 |
| DLL Control Register                       | 11-43 |
| PHY Control 1 Register                     | 11-44 |
| PHY Control 3 Register                     | 11-44 |

|                            |       |
|----------------------------|-------|
| PAD Control Register ..... | 11-46 |
|----------------------------|-------|

## Cyclic Redundancy Check (CRC)

|                                           |       |
|-------------------------------------------|-------|
| CRC Features .....                        | 12-1  |
| CRC Functional Description .....          | 12-2  |
| CRC Register List .....                   | 12-3  |
| ADSP-BF60x CRC Interrupt List .....       | 12-4  |
| CRC Definitions .....                     | 12-5  |
| CRC Block Diagram .....                   | 12-5  |
| Peripheral DMA Bus .....                  | 12-6  |
| MMR Access Bus .....                      | 12-6  |
| Compare Logic .....                       | 12-7  |
| Mirror Block .....                        | 12-7  |
| CRC Data FIFO .....                       | 12-7  |
| CRC Engine .....                          | 12-7  |
| DMA Request Generator .....               | 12-7  |
| CRC Architectural Concepts .....          | 12-7  |
| Lookup Table .....                        | 12-8  |
| Data Mirroring .....                      | 12-8  |
| FIFO Status and Data Requests .....       | 12-9  |
| CRC Operating Modes .....                 | 12-10 |
| Data Transfer Modes .....                 | 12-10 |
| Memory Scan Compute and Compare .....     | 12-11 |
| Memory Scan Data Verify .....             | 12-12 |
| Memory Transfer Compute and Compare ..... | 12-12 |
| Memory Transfer Data Fill Mode .....      | 12-13 |
| CRC Event Control .....                   | 12-13 |
| Interrupt Signals .....                   | 12-13 |
| CRC Programming Model .....               | 12-14 |
| CRC Mode Configuration .....              | 12-14 |
| Look-Up Table Generation .....            | 12-14 |

|                                                        |       |
|--------------------------------------------------------|-------|
| Core Driven Memory Scan Compute Compare Mode .....     | 12-15 |
| DMA Driven Memory Scan Compute Compare Mode .....      | 12-16 |
| Core Driven Memory Scan Data Verify Mode .....         | 12-18 |
| DMA Driven Memory Scan Data Verify Mode .....          | 12-20 |
| Core Driven Memory Transfer Compute Compare Mode ..... | 12-22 |
| DMA Driven Memory Transfer Compute Compare Mode .....  | 12-23 |
| DMA Driven Memory Transfer Data Fill Mode .....        | 12-26 |
| CRC Peripheral and DMA Channel List .....              | 12-27 |
| CRC Register Descriptions .....                        | 12-28 |
| Control Register .....                                 | 12-28 |
| Data Word Count Register .....                         | 12-33 |
| Data Word Count Reload Register .....                  | 12-33 |
| Data Compare Register .....                            | 12-34 |
| Fill Value Register .....                              | 12-35 |
| Data FIFO Register .....                               | 12-35 |
| Interrupt Enable Register .....                        | 12-36 |
| Interrupt Enable Set Register .....                    | 12-37 |
| Interrupt Enable Clear Register .....                  | 12-38 |
| Polynomial Register .....                              | 12-39 |
| Status Register .....                                  | 12-39 |
| Data Count Capture Register .....                      | 12-42 |
| CRC Final Result Register .....                        | 12-43 |
| CRC Current Result Register .....                      | 12-43 |
| Revision ID Register .....                             | 12-44 |

## Direct Memory Access (DMA)

|                                          |      |
|------------------------------------------|------|
| DMA Channel Features .....               | 13-1 |
| DMA Channel Functional Description ..... | 13-2 |
| DMA Register List .....                  | 13-3 |
| DMA Definitions .....                    | 13-4 |
| DMA Channel Block Diagram .....          | 13-5 |

|                                                           |       |
|-----------------------------------------------------------|-------|
| DMA Channel Architectural Concepts .....                  | 13-7  |
| DMA Channel SCB Interface .....                           | 13-7  |
| SCB Interface Signals .....                               | 13-7  |
| SCB Burst Transfers .....                                 | 13-8  |
| Data Address Alignment .....                              | 13-8  |
| Descriptor Set Address Alignment .....                    | 13-9  |
| DMA Channel Peripheral DMA Bus .....                      | 13-9  |
| Peripheral DMA Bus .....                                  | 13-9  |
| Peripheral DMA Bus Signals .....                          | 13-9  |
| DMA Channel Peripheral Control Commands .....             | 13-10 |
| DMA Channel Peripheral Control Command Restrictions ..... | 13-12 |
| DMA Channel MMR Access Bus .....                          | 13-13 |
| MMR Access Bus .....                                      | 13-14 |
| MMR Access Bus Signals .....                              | 13-14 |
| DMA Channel Operation Flow .....                          | 13-14 |
| DMA Channel Startup .....                                 | 13-14 |
| DMA Channel Refresh .....                                 | 13-16 |
| DMA Channel Work Unit Transitions .....                   | 13-17 |
| DMA Channel Transfer Termination and Shutdown .....       | 13-19 |
| DMA Channel Errors .....                                  | 13-21 |
| DMA Channel Memory DMA .....                              | 13-23 |
| MDMA Triggering .....                                     | 13-25 |
| DMA Operating Modes .....                                 | 13-25 |
| DMA Channel Register Based Flow Modes .....               | 13-25 |
| DMA Channel Stop Mode .....                               | 13-26 |
| Autobuffer Mode .....                                     | 13-26 |
| DMA Channel Descriptor Based Flow Modes .....             | 13-26 |
| DMA Channel Descriptor Array Mode .....                   | 13-27 |
| Descriptor Sets .....                                     | 13-27 |
| Minimum Startup Requirements .....                        | 13-27 |

|                                                                  |       |
|------------------------------------------------------------------|-------|
| DMA Channel Descriptor List Mode .....                           | 13-28 |
| Descriptor Sets .....                                            | 13-28 |
| Minimum Startup Requirements .....                               | 13-28 |
| DMA Channel Descriptor On Demand Array Mode .....                | 13-28 |
| Descriptor Sets .....                                            | 13-29 |
| Minimum Startup Requirements .....                               | 13-29 |
| DMA Channel Descriptor On Demand List Mode .....                 | 13-29 |
| Descriptor Sets .....                                            | 13-30 |
| Minimum Startup Requirements .....                               | 13-30 |
| DMA Channel Data Transfer Modes .....                            | 13-30 |
| DMA Channel Two-Dimensional DMA .....                            | 13-30 |
| DMA Channel Event Control .....                                  | 13-31 |
| Event Signals .....                                              | 13-32 |
| Work Unit State Events .....                                     | 13-33 |
| Peripheral Interrupt Request Events .....                        | 13-33 |
| Peripheral Data Request Events .....                             | 13-34 |
| DMA Channel Triggers .....                                       | 13-34 |
| Issuing Triggers .....                                           | 13-34 |
| Waiting For Triggers .....                                       | 13-35 |
| DMA Channel Programming Model .....                              | 13-36 |
| DMA Channel Mode Configuration .....                             | 13-36 |
| Register Based Linear Buffer Stop Flow Mode .....                | 13-36 |
| Register Based Autobuffer Flow Mode .....                        | 13-37 |
| Descriptor Array Flow Mode .....                                 | 13-38 |
| Descriptor List Flow Mode .....                                  | 13-39 |
| Register Based Memory to Memory Transfer in Stop Flow Mode ..... | 13-40 |
| DMA Channel Programming Concepts .....                           | 13-42 |
| Synchronization of Software and DMA .....                        | 13-42 |
| Interrupt and Trigger Event Based Synchronization .....          | 13-42 |
| Register Polling Based Synchronization .....                     | 13-42 |

|                                                              |       |
|--------------------------------------------------------------|-------|
| Descriptor Queues .....                                      | 13-43 |
| Queues Using Event Generation for Every Descriptor Set ..... | 13-43 |
| Queues Using Minimal Events .....                            | 13-44 |
| DMA Register Descriptions .....                              | 13-45 |
| Pointer to Next Initial Descriptor .....                     | 13-46 |
| Start Address of Current Buffer .....                        | 13-47 |
| Configuration Register .....                                 | 13-48 |
| Inner Loop Count Start Value .....                           | 13-56 |
| Inner Loop Address Increment .....                           | 13-57 |
| Outer Loop Count Start Value (2D only) .....                 | 13-58 |
| Outer Loop Address Increment (2D only) .....                 | 13-59 |
| Current Descriptor Pointer .....                             | 13-59 |
| Previous Initial Descriptor Pointer .....                    | 13-60 |
| Current Address .....                                        | 13-61 |
| Status Register .....                                        | 13-62 |
| Current Count(1D) or intra-row XCNT (2D) .....               | 13-66 |
| Current Row Count (2D only) .....                            | 13-67 |
| Bandwidth Limit Count .....                                  | 13-68 |
| Bandwidth Limit Count Current .....                          | 13-68 |
| Bandwidth Monitor Count .....                                | 13-69 |
| Bandwidth Monitor Count Current .....                        | 13-70 |
| DMA Channel List for ADSP-BF60x .....                        | 13-70 |

## General-Purpose Ports (PORT)

|                                    |      |
|------------------------------------|------|
| PORT Features .....                | 14-2 |
| PORT Functional Description .....  | 14-2 |
| Input Mode .....                   | 14-2 |
| Output Mode .....                  | 14-2 |
| Open-Drain Mode .....              | 14-3 |
| PORT Register List .....           | 14-3 |
| ADSP-BF60x PORT Muxing Table ..... | 14-4 |



|                                                  |       |
|--------------------------------------------------|-------|
| PINT Register List .....                         | 14-11 |
| ADSP-BF60x PINT Interrupt List .....             | 14-12 |
| ADSP-BF60x PINT Trigger List .....               | 14-12 |
| PORT Definitions .....                           | 14-13 |
| PORT Architectural Concepts .....                | 14-13 |
| External Interfaces .....                        | 14-13 |
| Internal Interfaces .....                        | 14-14 |
| PORT Event Control .....                         | 14-14 |
| PORT Interrupt Signals .....                     | 14-14 |
| PORT Programming Model .....                     | 14-16 |
| PORT Register Descriptions .....                 | 14-19 |
| Port x Function Enable Register .....            | 14-20 |
| Port x Function Enable Set Register .....        | 14-24 |
| Port x Function Enable Clear Register .....      | 14-27 |
| Port x GPIO Data Register .....                  | 14-30 |
| Port x GPIO Data Set Register .....              | 14-34 |
| Port x GPIO Data Clear Register .....            | 14-38 |
| Port x GPIO Direction Register .....             | 14-42 |
| Port x GPIO Direction Set Register .....         | 14-46 |
| Port x GPIO Direction Clear Register .....       | 14-50 |
| Port x GPIO Input Enable Register .....          | 14-54 |
| Port x GPIO Input Enable Set Register .....      | 14-58 |
| Port x GPIO Input Enable Clear Register .....    | 14-61 |
| Port x Multiplexer Control Register .....        | 14-65 |
| Port x GPIO Input Enable Toggle Register .....   | 14-67 |
| Port x GPIO Polarity Invert Register .....       | 14-71 |
| Port x GPIO Polarity Invert Set Register .....   | 14-76 |
| Port x GPIO Polarity Invert Clear Register ..... | 14-79 |
| Port x GPIO Lock Register .....                  | 14-83 |
| Port x GPIO Revision ID .....                    | 14-85 |

|                                  |        |
|----------------------------------|--------|
| PINT Register Descriptions ..... | 14-86  |
| Pint Mask Set Register .....     | 14-86  |
| Pint Mask Clear Register .....   | 14-90  |
| Pint Request Register .....      | 14-94  |
| Pint Assign Register .....       | 14-98  |
| Pint Edge Set Register .....     | 14-99  |
| Pint Edge Clear Register .....   | 14-103 |
| Pint Invert Set Register .....   | 14-107 |
| Pint Invert Clear Register ..... | 14-111 |
| Pint Pinstate Register .....     | 14-115 |
| Pint Latch Register .....        | 14-119 |

## General-Purpose Timer (TIMER)

|                                                              |       |
|--------------------------------------------------------------|-------|
| TIMER Features .....                                         | 15-1  |
| TIMER Register List .....                                    | 15-2  |
| ADSP-BF60x TIMER Interrupt List .....                        | 15-3  |
| ADSP-BF60x TIMER Trigger List .....                          | 15-3  |
| TIMER Internal Interface .....                               | 15-5  |
| TIMER External Interface .....                               | 15-5  |
| TIMER General Operation .....                                | 15-5  |
| Timer Period, Width and Delay Register Interaction .....     | 15-6  |
| TIMER Programming Concepts .....                             | 15-8  |
| Setting Up Constantly Changing Timer Conditions .....        | 15-8  |
| Configuring, Enabling and Disabling One or More Timers ..... | 15-8  |
| Configuring Timer Data and Status Interrupts .....           | 15-9  |
| Using the Timer Broadcast Feature .....                      | 15-9  |
| Timer Single-Pulse PWMOUT Mode .....                         | 15-9  |
| Timer Continuous PWMOUT Mode .....                           | 15-10 |
| TIMER Width Capture (WIDCAP) Mode .....                      | 15-11 |
| GP Timer Width Capture Mode Overflow .....                   | 15-15 |
| TIMER Windowed Watchdog (WATCHDOG) Modes .....               | 15-17 |

|                                           |       |
|-------------------------------------------|-------|
| Timer Windowed Watchdog Width Mode .....  | 15-18 |
| Timer Windowed Watchdog Period Mode ..... | 15-20 |
| TIMER Pin Interrupt (PININT) Mode .....   | 15-21 |
| TIMER External Clock (EXTCLK) Mode .....  | 15-22 |
| Timer Illegal States .....                | 15-23 |
| Continuous PWMOUT Mode .....              | 15-24 |
| Single Pulse PWMOUT Mode .....            | 15-25 |
| WID CAP Mode .....                        | 15-26 |
| EXTCLK Mode .....                         | 15-26 |
| WATCHDOG Events .....                     | 15-27 |
| TIMER Register Descriptions .....         | 15-28 |
| Revision ID Register .....                | 15-29 |
| Run Register .....                        | 15-29 |
| Run Set Register .....                    | 15-30 |
| Run Clear Register .....                  | 15-31 |
| Stop Configuration Register .....         | 15-32 |
| Stop Configuration Set Register .....     | 15-32 |
| Stop Configuration Clear Register .....   | 15-33 |
| Data Interrupt Mask Register .....        | 15-34 |
| Status Interrupt Mask Register .....      | 15-34 |
| Trigger Master Mask Register .....        | 15-35 |
| Trigger Slave Enable Register .....       | 15-35 |
| Data Interrupt Latch Register .....       | 15-36 |
| Status Interrupt Latch Register .....     | 15-37 |
| Error Type Status Register .....          | 15-37 |
| Broadcast Period Register .....           | 15-40 |
| Broadcast Width Register .....            | 15-41 |
| Broadcast Delay Register .....            | 15-41 |
| Timer n Configuration Register .....      | 15-42 |
| Timer n Counter Register .....            | 15-47 |

|                               |       |
|-------------------------------|-------|
| Timer n Period Register ..... | 15-48 |
| Timer n Width Register .....  | 15-49 |
| Timer n Delay Register .....  | 15-49 |

## Watchdog Timer (WDOG)

|                                             |      |
|---------------------------------------------|------|
| WDOG Features .....                         | 16-1 |
| Watchdog Timer Functional Description ..... | 16-1 |
| WDOG Register List .....                    | 16-2 |
| ADSP-BF60x WDOG Interrupt List .....        | 16-2 |
| WDOG Block Diagram .....                    | 16-2 |
| Internal Interface .....                    | 16-3 |
| External Interface .....                    | 16-3 |
| WDOG Configuration .....                    | 16-3 |
| WDOG Register Descriptions .....            | 16-4 |
| Control Register .....                      | 16-4 |
| Count Register .....                        | 16-5 |
| Watchdog Timer Status Register .....        | 16-6 |

## General-Purpose Counter (CNT)

|                                     |      |
|-------------------------------------|------|
| CNT Features .....                  | 17-1 |
| CNT Functional Description .....    | 17-1 |
| CNT Register List .....             | 17-2 |
| ADSP-BF60x CNT Interrupt List ..... | 17-3 |
| ADSP-BF60x CNT Trigger List .....   | 17-3 |
| CNT Operating Modes .....           | 17-4 |
| Quadrature Encoder Mode .....       | 17-4 |
| Binary Encoder Mode .....           | 17-4 |
| Up/Down Counter Mode .....          | 17-5 |
| Direction Counter Mode .....        | 17-5 |
| Timed Direction Mode .....          | 17-5 |
| CNT Event Control .....             | 17-5 |

|                                                                 |       |
|-----------------------------------------------------------------|-------|
| Illegal Gray/Binary Code Events .....                           | 17-6  |
| Up/Down Count Events .....                                      | 17-6  |
| Zero-Count Events .....                                         | 17-6  |
| Overflow Events .....                                           | 17-6  |
| Boundary Match Events .....                                     | 17-7  |
| Zero Marker Events .....                                        | 17-7  |
| CNT Programming Model .....                                     | 17-7  |
| CNT General Programming Flow .....                              | 17-7  |
| CNT Mode Configuration .....                                    | 17-7  |
| Configuring GP Counter Push-button Operation .....              | 17-8  |
| Configuring Zero-Marker-Zeros-Counter Mode .....                | 17-8  |
| Configuring Zero-Marker-Error Mode .....                        | 17-8  |
| Configuring Zero-Once Mode .....                                | 17-9  |
| Configuring Boundary Auto-Extend Mode .....                     | 17-9  |
| Configuring Boundary Capture Mode .....                         | 17-9  |
| Configuring Boundary Compare and Boundary Zero Modes .....      | 17-10 |
| CNT Programming Concepts .....                                  | 17-10 |
| CNT Input Noise Filtering .....                                 | 17-10 |
| Capturing Counter Interval and CNT_CNTR Read Timing .....       | 17-11 |
| Capturing Time Interval Between Successive Counter Events ..... | 17-13 |
| CNT Register Descriptions .....                                 | 17-14 |
| Configuration Register .....                                    | 17-14 |
| Interrupt Mask Register .....                                   | 17-17 |
| Status Register .....                                           | 17-20 |
| Command Register .....                                          | 17-21 |
| Debounce Register .....                                         | 17-23 |
| Counter Register .....                                          | 17-26 |
| Maximum Count Register .....                                    | 17-26 |
| Minimum Count Register .....                                    | 17-27 |

## Pulse-Width Modulator (PWM)

|                                                                          |       |
|--------------------------------------------------------------------------|-------|
| PWM Features .....                                                       | 18-1  |
| PWM Register List .....                                                  | 18-1  |
| ADSP-BF60x PWM Interrupt List .....                                      | 18-3  |
| ADSP-BF60x PWM Trigger List .....                                        | 18-4  |
| Functional Description .....                                             | 18-4  |
| Architectural Concepts .....                                             | 18-5  |
| Block Diagram .....                                                      | 18-5  |
| Timer Units .....                                                        | 18-6  |
| PWM Switching Frequency (PWM_TM) Register .....                          | 18-7  |
| Timer Unit Operation .....                                               | 18-7  |
| Phase Offset Control .....                                               | 18-9  |
| Channel Timing Control Unit .....                                        | 18-13 |
| Channel Control .....                                                    | 18-13 |
| Pulse Positioning and Duty Cycle Registers .....                         | 18-14 |
| Duty Cycle and Pulse Positioning Control .....                           | 18-14 |
| Channel Low Side Output Dependent Operation Mode and Dead-Time .....     | 18-15 |
| Channel High Side and Low Side Outputs, Independent Operation Mode ..... | 18-17 |
| Switched Reluctance Motors Application .....                             | 18-19 |
| Switching Dead Time (PWM_DT) Register .....                              | 18-20 |
| Duty Cycle with Dead-Time Control: Calculations for PULSEMODE 00 .....   | 18-20 |
| Special Consideration for PWM Operation in Over-Modulation .....         | 18-22 |
| Output Disable and Cross-Over Functions .....                            | 18-23 |
| Brushless DC Motor (Electronically Commutated Motor) Control .....       | 18-24 |
| Gate Drive Unit .....                                                    | 18-25 |
| Output Control Feature Precedence .....                                  | 18-26 |
| Sync Operation .....                                                     | 18-26 |
| Internal PWM SYNC Generation .....                                       | 18-27 |
| External PWM SYNC Generation .....                                       | 18-27 |
| Event Control .....                                                      | 18-27 |
| Trip Control Unit .....                                                  | 18-28 |

|                                                          |       |
|----------------------------------------------------------|-------|
| Programming Model .....                                  | 18-30 |
| Programming Model for 3-Phase AC Motor Control .....     | 18-30 |
| System Parameters .....                                  | 18-31 |
| System State Sequencing .....                            | 18-32 |
| PWM Initialization for Motor Control .....               | 18-32 |
| PWM Enable for Motor Control .....                       | 18-35 |
| PWM Response to Sync Interrupt for Motor Control .....   | 18-36 |
| PWM Disable (and Stop the Motor) for Motor Control ..... | 18-37 |
| PWM Register Descriptions .....                          | 18-38 |
| Control Register .....                                   | 18-39 |
| Channel Config Register .....                            | 18-43 |
| Trip Config Register .....                               | 18-51 |
| Status Register .....                                    | 18-55 |
| Interrupt Mask Register .....                            | 18-61 |
| Interrupt Latch Register .....                           | 18-64 |
| Chop Configuration Register .....                        | 18-66 |
| Dead Time Register .....                                 | 18-67 |
| Sync Pulse Width Register .....                          | 18-68 |
| Timer 0 Period Register .....                            | 18-69 |
| Timer 1 Period Register .....                            | 18-69 |
| Timer 2 Period Register .....                            | 18-70 |
| Timer 3 Period Register .....                            | 18-71 |
| Timer 4 Period Register .....                            | 18-72 |
| Channel A Delay Register .....                           | 18-73 |
| Channel B Delay Register .....                           | 18-74 |
| Channel C Delay Register .....                           | 18-75 |
| Channel D Delay Register .....                           | 18-75 |
| Channel A Control Register .....                         | 18-76 |
| Channel A-High Duty-0 Register .....                     | 18-79 |
| Channel A-High Duty-1 Register .....                     | 18-80 |

|                                            |        |
|--------------------------------------------|--------|
| Channel A-Low Duty-0 Register .....        | 18-81  |
| Channel A-Low Duty-1 Register .....        | 18-82  |
| Channel B Control Register .....           | 18-83  |
| Channel B-High Duty-0 Register .....       | 18-86  |
| Channel B-High Duty-1 Register .....       | 18-87  |
| Channel B-Low Duty-0 Register .....        | 18-88  |
| Channel B-Low Duty-1 Register .....        | 18-89  |
| Channel C Control Register .....           | 18-89  |
| Channel C-High Pulse Duty Register 0 ..... | 18-92  |
| Channel C-High Pulse Duty Register 1 ..... | 18-93  |
| Channel C-Low Pulse Duty Register 0 .....  | 18-94  |
| Channel C-Low Duty-1 Register .....        | 18-95  |
| Channel D Control Register .....           | 18-96  |
| Channel D-High Duty-0 Register .....       | 18-98  |
| Channel D-High Pulse Duty Register 1 ..... | 18-99  |
| Channel D-Low Pulse Duty Register 0 .....  | 18-100 |
| Channel D-Low Pulse Duty Register 1 .....  | 18-101 |

## Universal Async Rx/Tx (UART)

|                                      |      |
|--------------------------------------|------|
| UART Features .....                  | 19-1 |
| UART Functional Description .....    | 19-3 |
| UART Register List .....             | 19-3 |
| ADSP-BF60x UART Interrupt List ..... | 19-4 |
| ADSP-BF60x UART Trigger List .....   | 19-4 |
| UART Block Diagram .....             | 19-5 |
| UART Architectural Concepts .....    | 19-5 |
| Internal Interface .....             | 19-6 |
| External Interface .....             | 19-6 |
| Hardware Flow Control .....          | 19-6 |
| UART Bit Rate Generation .....       | 19-7 |
| Autobaud Detection .....             | 19-8 |



|                                                                      |       |
|----------------------------------------------------------------------|-------|
| UART Debug Features .....                                            | 19-10 |
| UART Operating Modes .....                                           | 19-10 |
| UART Mode .....                                                      | 19-11 |
| IrDA SIR Mode .....                                                  | 19-11 |
| MDB Mode .....                                                       | 19-11 |
| UART Data Transfer Modes .....                                       | 19-12 |
| UART Mode Transmit Operation (Core) .....                            | 19-12 |
| UART Mode Break Command and Inter-Frame Gap Transmit Operation ..... | 19-12 |
| UART Mode Receive Operation (Core) .....                             | 19-13 |
| IrDA Transmit Operation .....                                        | 19-14 |
| IrDA Receive Operation .....                                         | 19-14 |
| MDB Transmit Operation .....                                         | 19-15 |
| MDB Receive Operation .....                                          | 19-16 |
| DMA Modes .....                                                      | 19-16 |
| Mixing DMA and Core Modes .....                                      | 19-17 |
| Setting Up Hardware Flow Control .....                               | 19-17 |
| UART Event Control .....                                             | 19-18 |
| Interrupt Masks .....                                                | 19-18 |
| Interrupt Servicing .....                                            | 19-18 |
| Transmit Interrupts .....                                            | 19-19 |
| Receive Interrupts .....                                             | 19-20 |
| Status Interrupts .....                                              | 19-21 |
| Multi-Drop Bus Events .....                                          | 19-22 |
| UART Programming Model .....                                         | 19-22 |
| Detecting Autobaud .....                                             | 19-22 |
| Using Common Initialization Steps .....                              | 19-23 |
| Using Core Transfers .....                                           | 19-23 |
| Using DMA Transfers .....                                            | 19-23 |
| Using Interrupts .....                                               | 19-23 |
| Setting Up Hardware Flow Control .....                               | 19-24 |

|                                              |       |
|----------------------------------------------|-------|
| UART Register Descriptions .....             | 19-24 |
| Revision ID Register .....                   | 19-25 |
| Control Register .....                       | 19-25 |
| Status Register .....                        | 19-33 |
| Scratch Register .....                       | 19-39 |
| Clock Rate Register .....                    | 19-39 |
| Interrupt Mask Register .....                | 19-40 |
| Interrupt Mask Set Register .....            | 19-45 |
| Interrupt Mask Clear Register .....          | 19-47 |
| Receive Buffer Register .....                | 19-49 |
| Transmit Hold Register .....                 | 19-50 |
| Transmit Address/Insert Pulse Register ..... | 19-50 |
| Transmit Shift Register .....                | 19-51 |
| Receive Shift Register .....                 | 19-51 |
| Transmit Counter Register .....              | 19-52 |
| Receive Counter Register .....               | 19-53 |

## 2-Wire Interface (TWI)

|                                            |      |
|--------------------------------------------|------|
| TWI Features .....                         | 20-1 |
| TWI Functional Description .....           | 20-2 |
| TWI Register List .....                    | 20-2 |
| ADSP-BF60x TWI Interrupt List .....        | 20-3 |
| TWI Block Diagram .....                    | 20-3 |
| TWI External Interface .....               | 20-4 |
| Serial Clock Signal (SCL) .....            | 20-4 |
| Serial Data Signal (SDA) .....             | 20-5 |
| TWI Internal Interface .....               | 20-5 |
| TWI Architectural Concepts .....           | 20-5 |
| TWI Protocol .....                         | 20-5 |
| Clock Generation and Synchronization ..... | 20-6 |
| Bus Arbitration .....                      | 20-7 |

|                                              |       |
|----------------------------------------------|-------|
| Start and Stop Conditions .....              | 20-7  |
| General Call Support .....                   | 20-8  |
| Fast Mode .....                              | 20-8  |
| TWI Operating Modes .....                    | 20-8  |
| Repeated Start .....                         | 20-8  |
| Transmit Receive Repeated Start .....        | 20-8  |
| Receive Transmit Repeated Start .....        | 20-9  |
| Clock Stretching .....                       | 20-9  |
| Clock Stretching During FIFO Underflow ..... | 20-10 |
| Clock Stretching During FIFO Overflow .....  | 20-10 |
| Clock Stretching During Repeated Start ..... | 20-11 |
| TWI Programming Model .....                  | 20-12 |
| TWI General Setup .....                      | 20-13 |
| Slave Mode .....                             | 20-13 |
| Master Mode Clock Setup .....                | 20-15 |
| Master Mode Transmit .....                   | 20-15 |
| Master Mode Receive .....                    | 20-16 |
| Master Mode Program Flow .....               | 20-17 |
| TWI Register Descriptions .....              | 20-17 |
| SCL Clock Divider Register .....             | 20-18 |
| Control Register .....                       | 20-19 |
| Slave Mode Control Register .....            | 20-21 |
| Slave Mode Status Register .....             | 20-22 |
| Slave Mode Address Register .....            | 20-23 |
| Master Mode Control Registers .....          | 20-24 |
| Master Mode Status Register .....            | 20-27 |
| Master Mode Address Register .....           | 20-30 |
| Interrupt Status Register .....              | 20-31 |
| Interrupt Mask Register .....                | 20-34 |
| FIFO Control Register .....                  | 20-36 |

|                                    |       |
|------------------------------------|-------|
| FIFO Status Register .....         | 20-38 |
| Tx Data Single-Byte Register ..... | 20-40 |
| Tx Data Double-Byte Register ..... | 20-40 |
| Rx Data Single-Byte Register ..... | 20-41 |
| Rx Data Double-Byte Register ..... | 20-41 |

## Controller Area Network (CAN)

|                                              |       |
|----------------------------------------------|-------|
| CAN Features .....                           | 21-1  |
| CAN Functional Description .....             | 21-2  |
| CAN Register List .....                      | 21-2  |
| ADSP-BF60x CAN Interrupt List .....          | 21-4  |
| CAN External Interface .....                 | 21-5  |
| ADSP-BF60x Specific External Interface ..... | 21-5  |
| CAN Architectural Concepts .....             | 21-5  |
| CAN Block Diagram .....                      | 21-6  |
| CAN Mailbox Control .....                    | 21-7  |
| CAN Protocol Fundamentals .....              | 21-8  |
| CAN Data Transfer Modes .....                | 21-10 |
| CAN Transmit Operation .....                 | 21-10 |
| Retransmission .....                         | 21-10 |
| Single-Shot Transmission .....               | 21-11 |
| Auto-Transmission .....                      | 21-11 |
| CAN Receive Operation .....                  | 21-12 |
| CAN Data Acceptance Filtering .....          | 21-14 |
| CAN Watchdog Mode .....                      | 21-15 |
| CAN Time Stamps .....                        | 21-15 |
| CAN Remote Frame Handling .....              | 21-16 |
| CAN Mailbox Disabling (Temporary) .....      | 21-17 |
| CAN Operating Modes .....                    | 21-17 |
| Bit Timing .....                             | 21-18 |
| CAN Low Power Features .....                 | 21-19 |

|                                                                |       |
|----------------------------------------------------------------|-------|
| CAN Built-In Suspend Mode .....                                | 21-19 |
| CAN Built-In Sleep Mode .....                                  | 21-20 |
| CAN Wakeup From Hibernate State .....                          | 21-20 |
| Soft Reset .....                                               | 21-20 |
| CAN Event Control .....                                        | 21-21 |
| CAN Interrupt Signals .....                                    | 21-21 |
| Mailbox Interrupts .....                                       | 21-21 |
| Global CAN Interrupt .....                                     | 21-21 |
| CAN Event Counter .....                                        | 21-23 |
| CAN Warnings and Errors .....                                  | 21-24 |
| Programmable Warning Limits .....                              | 21-24 |
| Error Handling .....                                           | 21-24 |
| Error Frames .....                                             | 21-25 |
| Error Levels .....                                             | 21-26 |
| CAN Debug and Test Modes .....                                 | 21-27 |
| CAN Register Descriptions .....                                | 21-29 |
| Mailbox Configuration 1 Register .....                         | 21-32 |
| Mailbox Direction 1 Register .....                             | 21-32 |
| Transmission Request Set 1 Register .....                      | 21-33 |
| Transmission Request Reset 1 Register .....                    | 21-33 |
| Transmission Acknowledge 1 Register .....                      | 21-34 |
| Abort Acknowledge 1 Register .....                             | 21-34 |
| Receive Message Pending 1 Register .....                       | 21-35 |
| Receive Message Lost 1 Register .....                          | 21-35 |
| Mailbox Transmit Interrupt Flag 1 Register .....               | 21-36 |
| Mailbox Receive Interrupt Flag 1 Register .....                | 21-37 |
| Mailbox Interrupt Mask 1 Register .....                        | 21-37 |
| Remote Frame Handling 1 Register .....                         | 21-38 |
| Overwrite Protection/Single Shot Transmission 1 Register ..... | 21-38 |
| Mailbox Configuration 2 Register .....                         | 21-39 |

|                                                                |       |
|----------------------------------------------------------------|-------|
| Mailbox Direction 2 Register .....                             | 21-40 |
| Transmission Request Set 2 Register .....                      | 21-40 |
| Transmission Request Reset 2 Register .....                    | 21-41 |
| Transmission Acknowledge 2 Register .....                      | 21-41 |
| Abort Acknowledge 2 Register .....                             | 21-42 |
| Receive Message Pending 2 Register .....                       | 21-42 |
| Receive Message Lost 2 Register .....                          | 21-43 |
| Mailbox Transmit Interrupt Flag 2 Register .....               | 21-43 |
| Mailbox Receive Interrupt Flag 2 Register .....                | 21-44 |
| Mailbox Interrupt Mask 2 Register .....                        | 21-44 |
| Remote Frame Handling 2 Register .....                         | 21-45 |
| Overwrite Protection/Single Shot Transmission 2 Register ..... | 21-45 |
| Clock Register .....                                           | 21-46 |
| Timing Register .....                                          | 21-47 |
| Debug Register .....                                           | 21-48 |
| Status Register .....                                          | 21-51 |
| Error Counter Register .....                                   | 21-53 |
| Global CAN Interrupt Status Register .....                     | 21-54 |
| Global CAN Interrupt Mask Register .....                       | 21-58 |
| Global CAN Interrupt Flag Register .....                       | 21-60 |
| CAN Master Control Register .....                              | 21-63 |
| Interrupt Pending Register .....                               | 21-65 |
| Temporary Mailbox Disable Register .....                       | 21-67 |
| Error Counter Warning Level Register .....                     | 21-68 |
| Error Status Register .....                                    | 21-69 |
| Universal Counter Register .....                               | 21-71 |
| Universal Counter Reload/Capture Register .....                | 21-72 |
| Universal Counter Configuration Mode Register .....            | 21-73 |
| Acceptance Mask (L) Register .....                             | 21-75 |
| Acceptance Mask (H) Register .....                             | 21-76 |

|                                  |       |
|----------------------------------|-------|
| Mailbox Word 0 Register .....    | 21-77 |
| Mailbox Word 1 Register .....    | 21-78 |
| Mailbox Word 2 Register .....    | 21-78 |
| Mailbox Word 3 Register .....    | 21-79 |
| Mailbox Length Register .....    | 21-80 |
| Mailbox Timestamp Register ..... | 21-80 |
| Mailbox ID 0 Register .....      | 21-81 |
| Mailbox ID 1 Register .....      | 21-81 |

## Universal Serial Bus (USB)

|                                                         |       |
|---------------------------------------------------------|-------|
| USB Features .....                                      | 22-1  |
| USB Functional Description .....                        | 22-2  |
| USB Interface Overview .....                            | 22-2  |
| Multi-point Support .....                               | 22-3  |
| USB On-Chip Bus Interfaces .....                        | 22-3  |
| USB FIFO Configuration .....                            | 22-4  |
| USB Power and Clocking .....                            | 22-4  |
| USB UTMI Interface .....                                | 22-5  |
| USB Register List .....                                 | 22-5  |
| ADSP-BF60x USB Interrupt List .....                     | 22-8  |
| ADSP-BF60x USB Trigger List .....                       | 22-9  |
| USB Block Diagram .....                                 | 22-9  |
| USB Definitions .....                                   | 22-11 |
| USB References .....                                    | 22-12 |
| USB Operating Modes .....                               | 22-12 |
| USB Peripheral Mode .....                               | 22-13 |
| Endpoint Setup .....                                    | 22-13 |
| IN Transactions as a Peripheral .....                   | 22-14 |
| OUT Transactions as a Peripheral .....                  | 22-15 |
| High-Bandwidth Isochronous/Interrupt Transactions ..... | 22-16 |
| High Bandwidth Isochronous/Interrupt IN Endpoints ..... | 22-16 |

|                                                                           |       |
|---------------------------------------------------------------------------|-------|
| High Bandwidth Isochronous/Interrupt OUT Endpoints .....                  | 22-18 |
| Peripheral Transfer Work Flows .....                                      | 22-19 |
| Control Transactions as a Peripheral .....                                | 22-20 |
| Write Requests .....                                                      | 22-20 |
| Read Requests .....                                                       | 22-21 |
| Zero Data Requests .....                                                  | 22-22 |
| ENDPOINT 0 States .....                                                   | 22-22 |
| Endpoint 0 Service Routine as Peripheral .....                            | 22-23 |
| Peripheral Mode, Bulk IN, Transfer Size Known .....                       | 22-28 |
| Peripheral Mode, Bulk IN, Transfer Size Unknown .....                     | 22-29 |
| Peripheral Mode, ISO IN, Small MaxPktSize .....                           | 22-29 |
| Peripheral Mode, ISO IN, Large MaxPktSize .....                           | 22-30 |
| Peripheral Mode, Bulk OUT, Transfer Size Known .....                      | 22-30 |
| Peripheral Mode, Bulk OUT, Transfer Size Unknown .....                    | 22-31 |
| Peripheral Mode, ISO OUT, Small MaxPktSize .....                          | 22-31 |
| Peripheral Mode, ISO OUT, Large MaxPktSize .....                          | 22-32 |
| Peripheral Mode Suspend .....                                             | 22-32 |
| Start-of-frame (SOF) Packets .....                                        | 22-32 |
| Soft Connect/Soft Disconnect .....                                        | 22-33 |
| Error Handling As a Peripheral .....                                      | 22-33 |
| Stalls Issued to Control Transfers .....                                  | 22-34 |
| Zero Length OUT Data Packets in Control Transfers .....                   | 22-34 |
| USB Host Mode .....                                                       | 22-34 |
| Transaction Scheduling .....                                              | 22-35 |
| Endpoint Setup and Data Transfer .....                                    | 22-35 |
| Control Transaction as a Host .....                                       | 22-35 |
| Setup Phase as a Host .....                                               | 22-36 |
| IN Data Phase as a Host .....                                             | 22-37 |
| OUT Data as a Host (Control) .....                                        | 22-37 |
| IN Status Phase as a Host (Following SETUP Phase or OUT Data Phase) ..... | 22-38 |



|                                                                                       |       |
|---------------------------------------------------------------------------------------|-------|
| OUT Status Phase as a Host (following IN Data Phase) .....                            | 22-39 |
| Host IN Transactions .....                                                            | 22-39 |
| Host OUT Transactions .....                                                           | 22-40 |
| Multi-point Support .....                                                             | 22-40 |
| Allocating Devices to Endpoints .....                                                 | 22-40 |
| Multi-point Operation .....                                                           | 22-41 |
| Multi-point Bandwidth Considerations .....                                            | 22-42 |
| Babble Interrupt .....                                                                | 22-42 |
| VBUS Events .....                                                                     | 22-43 |
| Actions as an “A” Device .....                                                        | 22-43 |
| Actions as a “B” Device .....                                                         | 22-44 |
| Host Mode Reset .....                                                                 | 22-44 |
| Host Mode Suspend .....                                                               | 22-44 |
| Suspending and Resuming the Controller .....                                          | 22-45 |
| Suspend/Resume By Inactivity On The USB Bus (L0 To L2 State) In Peripheral Mode ..... | 22-45 |
| Suspend/Resume By Inactivity On The USB Bus (L0 To L2 State) In Host Mode .....       | 22-46 |
| Suspend/Resume By an LPM Transaction (L0 To L1 State) In Peripheral Mode .....        | 22-46 |
| Suspend/Resume by an LPM Transaction (L0 to L1 State) in Host Mode .....              | 22-48 |
| USB Event Control .....                                                               | 22-49 |
| USB Interrupt Signals .....                                                           | 22-49 |
| Interrupt Handling .....                                                              | 22-51 |
| USB Reset Signals .....                                                               | 22-53 |
| USB Reset in Peripheral Mode .....                                                    | 22-53 |
| USB Reset in Host Mode .....                                                          | 22-53 |
| USB Programming Model .....                                                           | 22-54 |
| Peripheral Mode Flow Charts .....                                                     | 22-54 |
| Host Mode Flow Charts .....                                                           | 22-62 |
| DMA Mode Flow Charts .....                                                            | 22-71 |
| OTG Session Request .....                                                             | 22-76 |
| Starting a Session .....                                                              | 22-77 |

|                                                |        |
|------------------------------------------------|--------|
| Detecting Activity .....                       | 22-77  |
| Host Negotiation/Configuration .....           | 22-78  |
| Wakeup from Hibernate State .....              | 22-78  |
| Data Transfer .....                            | 22-79  |
| Loading/Unloading Packets from Endpoints ..... | 22-80  |
| DMA Master Channels .....                      | 22-80  |
| DMA Bus Cycles .....                           | 22-81  |
| Transferring Packets Using DMA .....           | 22-83  |
| Individual Packet: RX Endpoint .....           | 22-83  |
| Individual Packet: TX Endpoint .....           | 22-83  |
| Multiple Packets: RX Endpoint .....            | 22-84  |
| Multiple Packets: TX Endpoints .....           | 22-85  |
| USB Register Descriptions .....                | 22-85  |
| Function Address Register .....                | 22-88  |
| Power and Device Control Register .....        | 22-89  |
| Transmit Interrupt Register .....              | 22-92  |
| Receive Interrupt Register .....               | 22-95  |
| Transmit Interrupt Enable Register .....       | 22-98  |
| Receive Interrupt Enable Register .....        | 22-102 |
| Common Interrupts Register .....               | 22-104 |
| Common Interrupts Enable Register .....        | 22-107 |
| Frame Number Register .....                    | 22-109 |
| Index Register .....                           | 22-110 |
| Testmode Register .....                        | 22-111 |
| FIFO Byte (8-Bit) Register .....               | 22-112 |
| FIFO Half-Word (16-Bit) Register .....         | 22-113 |
| FIFO Word (32-Bit) Register .....              | 22-113 |
| Device Control Register .....                  | 22-114 |
| Transmit FIFO Size Register .....              | 22-116 |
| Receive FIFO Size Register .....               | 22-118 |

|                                                                   |        |
|-------------------------------------------------------------------|--------|
| Transmit FIFO Address Register .....                              | 22-120 |
| Receive FIFO Address Register .....                               | 22-121 |
| Endpoint Information Register .....                               | 22-122 |
| RAM Information Register .....                                    | 22-123 |
| Link Information Register .....                                   | 22-123 |
| VBUS Pulse Length Register .....                                  | 22-124 |
| High-Speed EOF 1 Register .....                                   | 22-125 |
| Full-Speed EOF 1 Register .....                                   | 22-125 |
| Low-Speed EOF 1 Register .....                                    | 22-126 |
| Software Reset Register .....                                     | 22-126 |
| MPn Transmit Function Address Register .....                      | 22-127 |
| MPn Transmit Hub Address Register .....                           | 22-128 |
| MPn Transmit Hub Port Register .....                              | 22-129 |
| MPn Receive Function Address Register .....                       | 22-130 |
| MPn Receive Hub Address Register .....                            | 22-130 |
| MPn Receive Hub Port Register .....                               | 22-131 |
| EPn Transmit Maximum Packet Length Register .....                 | 22-131 |
| EP0 Configuration and Status (Host) Register .....                | 22-132 |
| EP0 Configuration and Status (Peripheral) Register .....          | 22-136 |
| EPn Transmit Configuration and Status (Host) Register .....       | 22-139 |
| EPn Transmit Configuration and Status (Peripheral) Register ..... | 22-144 |
| EPn Receive Maximum Packet Length Register .....                  | 22-149 |
| EPn Receive Configuration and Status (Host) Register .....        | 22-150 |
| EPn Receive Configuration and Status (Peripheral) Register .....  | 22-155 |
| EP0 Number of Received Bytes Register .....                       | 22-161 |
| EPn Number of Bytes Received Register .....                       | 22-161 |
| EPn Transmit Type Register .....                                  | 22-162 |
| EP0 Connection Type Register .....                                | 22-164 |
| EP0 NAK Limit Register .....                                      | 22-165 |
| EPn Transmit Polling Interval Register .....                      | 22-166 |

|                                                 |        |
|-------------------------------------------------|--------|
| EPn Receive Type Register .....                 | 22-167 |
| EPn Receive Polling Interval Register .....     | 22-169 |
| EP0 Configuration Information Register .....    | 22-170 |
| DMA Interrupt Register .....                    | 22-173 |
| DMA Channel n Control Register .....            | 22-175 |
| DMA Channel n Address Register .....            | 22-178 |
| DMA Channel n Count Register .....              | 22-179 |
| EPn Request Packet Count Register .....         | 22-180 |
| Chirp Timeout Register .....                    | 22-180 |
| Host High Speed Return to Normal Register ..... | 22-181 |
| High Speed Timeout Register .....               | 22-182 |
| LPM Attribute Register .....                    | 22-182 |
| LPM Control Register .....                      | 22-184 |
| LPM Interrupt Enable Register .....             | 22-186 |
| LPM Interrupt Status Register .....             | 22-188 |
| LPM Function Address Register .....             | 22-191 |
| VBUS Control Register .....                     | 22-192 |
| Battery Charging Control Register .....         | 22-193 |
| PHY Control Register .....                      | 22-194 |
| PLL and Oscillator Control Register .....       | 22-195 |

## Ethernet Media Access Controller (EMAC)

|                                         |       |
|-----------------------------------------|-------|
| Overview .....                          | 23-1  |
| Introduction .....                      | 23-1  |
| EMAC Features .....                     | 23-2  |
| EMAC Functional Description .....       | 23-2  |
| EMAC Register List .....                | 23-3  |
| ADSP-BF60x EMAC Interrupt List .....    | 23-9  |
| EMAC Definitions .....                  | 23-10 |
| EMAC Block Diagram and Interfaces ..... | 23-10 |
| EMAC CORE Sub-blocks .....              | 23-12 |

|                                                 |       |
|-------------------------------------------------|-------|
| PHY Interface .....                             | 23-14 |
| EMAC Clock Sources .....                        | 23-15 |
| EMAC Architectural Concepts .....               | 23-16 |
| EMAC System Crossbar Interface (EMAC SCB) ..... | 23-16 |
| Priority of SCB Requests .....                  | 23-17 |
| SCB Interface Programming Options .....         | 23-18 |
| DMA Bursts with SCB Interface .....             | 23-19 |
| SCB Bus Transaction Status .....                | 23-20 |
| Fatal Bus Error .....                           | 23-20 |
| EMAC DMA Controller (EMAC DMA) .....            | 23-20 |
| DMA Related Registers .....                     | 23-21 |
| DMA Descriptors .....                           | 23-23 |
| Transmit Descriptor .....                       | 23-24 |
| Receive Descriptor .....                        | 23-29 |
| EMAC DMA Transmit Process .....                 | 23-36 |
| Default (Non-OSF) Mode .....                    | 23-36 |
| OSF Mode Enabled .....                          | 23-37 |
| Transmit Frame Processing .....                 | 23-38 |
| Transmit Polling Suspended .....                | 23-38 |
| EMAC DMA Receive Process .....                  | 23-39 |
| Receive Frame Processing .....                  | 23-40 |
| Receive Descriptor Acquisition .....            | 23-40 |
| Receive Process Suspended .....                 | 23-41 |
| OWN Bit (Ownership) Semaphore .....             | 23-41 |
| Application Data Buffer Alignment .....         | 23-42 |
| Buffer Size Calculations .....                  | 23-42 |
| EMAC FIFO Layer (EMAC MFL) .....                | 23-43 |
| FIFO Size .....                                 | 23-43 |
| FIFO Layer Transmit Path .....                  | 23-43 |
| FIFO Layer Receive Path .....                   | 23-44 |

|                                                         |       |
|---------------------------------------------------------|-------|
| EMAC CORE .....                                         | 23-45 |
| EMAC CORE Transmission Engine .....                     | 23-48 |
| EMAC CORE Reception Engine .....                        | 23-52 |
| EMAC Station Management Interface (SMI) .....           | 23-59 |
| MDC Clock Frequency .....                               | 23-60 |
| SMI Write Operation .....                               | 23-62 |
| SMI Read Operation .....                                | 23-62 |
| EMAC Management Counters (MMC) .....                    | 23-63 |
| MMC Receive Interrupt Register .....                    | 23-64 |
| MMC Transmit Interrupt Register .....                   | 23-64 |
| MMC Receive Checksum Offload Interrupt Register .....   | 23-65 |
| EMAC Precision Time Protocol (PTP) Engine .....         | 23-65 |
| Introduction to IEEE1588 and PTP Engine .....           | 23-65 |
| Block Diagram .....                                     | 23-70 |
| PTP Module Clock .....                                  | 23-71 |
| Timestamp Module .....                                  | 23-72 |
| System Time .....                                       | 23-79 |
| Target Time Trigger (Alarm) .....                       | 23-82 |
| Pulse-Per-Second (PPS) .....                            | 23-82 |
| Interrupts related to PTP .....                         | 23-85 |
| EMAC Event Control .....                                | 23-86 |
| EMAC Interrupt Signals .....                            | 23-86 |
| PHYINT Interrupt Signal .....                           | 23-88 |
| EMAC Programming Model .....                            | 23-89 |
| EMAC Programming Steps .....                            | 23-89 |
| DMA Initialization .....                                | 23-89 |
| EMAC CORE Initialization .....                          | 23-90 |
| Performing Normal Transmit and Receive Operations ..... | 23-91 |
| Stopping and Starting Transfers .....                   | 23-92 |
| EMAC Interrupts and ISRs .....                          | 23-92 |

|                                                                   |        |
|-------------------------------------------------------------------|--------|
| Enabling Checksum for Transmit and Receive .....                  | 23-94  |
| Programming System Time Module .....                              | 23-94  |
| Programming PTP module for Frame detection and Timestamping ..... | 23-96  |
| Programming for Auxiliary Timestamps .....                        | 23-96  |
| Programming Pulse-Per-Second Output .....                         | 23-97  |
| EMAC Programming Concepts .....                                   | 23-98  |
| IEEE 802.3 Ethernet Packet Structure .....                        | 23-98  |
| Frame Size Statistics for Application Software .....              | 23-99  |
| Software Visualization of Programmable Packet Size .....          | 23-99  |
| Ethernet Packet Structure in C .....                              | 23-100 |
| DMA Descriptor Implementation in C .....                          | 23-100 |
| PTP Header Structure in C .....                                   | 23-101 |
| EMAC Register Descriptions .....                                  | 23-101 |
| MAC Configuration Register .....                                  | 23-107 |
| MAC Rx Frame Filter Register .....                                | 23-112 |
| Hash Table High Register .....                                    | 23-115 |
| Hash Table Low Register .....                                     | 23-116 |
| SMI Address Register .....                                        | 23-117 |
| SMI Data Register .....                                           | 23-119 |
| FLow Control Register .....                                       | 23-120 |
| VLAN Tag Register .....                                           | 23-121 |
| Debug Register .....                                              | 23-122 |
| Interrupt Status Register .....                                   | 23-125 |
| Interrupt Mask Register .....                                     | 23-127 |
| MAC Address 0 High Register .....                                 | 23-127 |
| MAC Address 0 Low Register .....                                  | 23-128 |
| MMC Control Register .....                                        | 23-129 |
| MMC Rx Interrupt Register .....                                   | 23-131 |
| MMC Tx Interrupt Register .....                                   | 23-134 |
| MMC Rx Interrupt Mask Register .....                              | 23-138 |

|                                                       |        |
|-------------------------------------------------------|--------|
| MMC TX Interrupt Mask Register .....                  | 23-142 |
| Tx OCT Count (Good/Bad) Register .....                | 23-146 |
| Tx Frame Count (Good/Bad) Register .....              | 23-147 |
| Tx Broadcast Frames (Good) Register .....             | 23-147 |
| Tx Multicast Frames (Good) Register .....             | 23-148 |
| Tx 64-Byte Frames (Good/Bad) Register .....           | 23-148 |
| Tx 65- to 127-Byte Frames (Good/Bad) Register .....   | 23-149 |
| Tx 128- to 255-Byte Frames (Good/Bad) Register .....  | 23-150 |
| Tx 256- to 511-Byte Frames (Good/Bad) Register .....  | 23-150 |
| Tx 512- to 1023-Byte Frames (Good/Bad) Register ..... | 23-151 |
| Tx 1024- to Max-Byte Frames (Good/Bad) Register ..... | 23-151 |
| Tx Unicast Frames (Good/Bad) Register .....           | 23-152 |
| Tx Multicast Frames (Good/Bad) Register .....         | 23-153 |
| Tx Broadcast Frames (Good/Bad) Register .....         | 23-153 |
| Tx Underflow Error Register .....                     | 23-154 |
| Tx Single Collision (Good) Register .....             | 23-154 |
| Tx Multiple Collision (Good) Register .....           | 23-155 |
| Tx Deferred Register .....                            | 23-156 |
| Tx Late Collision Register .....                      | 23-156 |
| Tx Excess Collision Register .....                    | 23-157 |
| Tx Carrier Error Register .....                       | 23-157 |
| Tx Octet Count (Good) Register .....                  | 23-158 |
| Tx Frame Count (Good) Register .....                  | 23-159 |
| Tx Excess Deferral Register .....                     | 23-159 |
| Tx Pause Frame Register .....                         | 23-160 |
| Tx VLAN Frames (Good) Register .....                  | 23-160 |
| Rx Frame Count (Good/Bad) Register .....              | 23-161 |
| Rx Octet Count (Good/Bad) Register .....              | 23-162 |
| Rx Octet Count (Good) Register .....                  | 23-162 |
| Rx Broadcast Frames (Good) Register .....             | 23-163 |



|                                                       |        |
|-------------------------------------------------------|--------|
| Rx Multicast Frames (Good) Register .....             | 23-163 |
| Rx CRC Error Register .....                           | 23-164 |
| Rx alignment Error Register .....                     | 23-165 |
| Rx Runt Error Register .....                          | 23-165 |
| Rx Jab Error Register .....                           | 23-166 |
| Rx Undersize (Good) Register .....                    | 23-166 |
| Rx Oversize (Good) Register .....                     | 23-167 |
| Rx 64-Byte Frames (Good/Bad) Register .....           | 23-168 |
| Rx 65- to 127-Byte Frames (Good/Bad) Register .....   | 23-168 |
| Rx 128- to 255-Byte Frames (Good/Bad) Register .....  | 23-169 |
| Rx 256- to 511-Byte Frames (Good/Bad) Register .....  | 23-169 |
| Rx 512- to 1023-Byte Frames (Good/Bad) Register ..... | 23-170 |
| Rx 1024- to Max-Byte Frames (Good/Bad) Register ..... | 23-171 |
| Rx Unicast Frames (Good) Register .....               | 23-171 |
| Rx Length Error Register .....                        | 23-172 |
| Rx Out Of Range Type Register .....                   | 23-172 |
| Rx Pause Frames Register .....                        | 23-173 |
| Rx FIFO Overflow Register .....                       | 23-174 |
| Rx VLAN Frames (Good/Bad) Register .....              | 23-174 |
| Rx Watch Dog Error Register .....                     | 23-175 |
| MMC IPC Rx Interrupt Mask Register .....              | 23-175 |
| MMC IPC Rx Interrupt Register .....                   | 23-180 |
| Rx IPv4 Datagrams (Good) Register .....               | 23-185 |
| Rx IPv4 Datagrams Header Errors Register .....        | 23-186 |
| Rx IPv4 Datagrams No Payload Frame Register .....     | 23-186 |
| Rx IPv4 Datagrams Fragmented Frames Register .....    | 23-187 |
| Rx IPv4 UDP Disabled Frames Register .....            | 23-188 |
| Rx IPv6 Datagrams Good Frames Register .....          | 23-188 |
| Rx IPv6 Datagrams Header Error Frames Register .....  | 23-189 |
| Rx IPv6 Datagrams No Payload Frames Register .....    | 23-189 |

|                                                    |        |
|----------------------------------------------------|--------|
| Rx UDP Good Frames Register .....                  | 23-190 |
| Rx UDP Error Frames Register .....                 | 23-191 |
| Rx TCP Good Frames Register .....                  | 23-191 |
| Rx TCP Error Frames Register .....                 | 23-192 |
| Rx ICMP Good Frames Register .....                 | 23-192 |
| Rx ICMP Error Frames Register .....                | 23-193 |
| Rx IPv4 Datagrams Good Octets Register .....       | 23-194 |
| Rx IPv4 Datagrams Header Errors Register .....     | 23-194 |
| Rx IPv4 Datagrams No Payload Octets Register ..... | 23-195 |
| Rx IPv4 Datagrams Fragmented Octets Register ..... | 23-195 |
| Rx IPv4 UDP Disabled Octets Register .....         | 23-196 |
| Rx IPv6 Good Octets Register .....                 | 23-197 |
| Rx IPv6 Header Errors Register .....               | 23-197 |
| Rx IPv6 No Payload Octets Register .....           | 23-198 |
| Rx UDP Good Octets Register .....                  | 23-198 |
| Rx UDP Error Octets Register .....                 | 23-199 |
| Rx TCP Good Octets Register .....                  | 23-200 |
| Rx TCP Error Octets Register .....                 | 23-200 |
| Rx ICMP Good Octets Register .....                 | 23-201 |
| Rx ICMP Error Octets Register .....                | 23-201 |
| Time Stamp Control Register .....                  | 23-202 |
| Time Stamp Sub Second Increment Register .....     | 23-208 |
| Time Stamp Low Seconds Register .....              | 23-209 |
| Time Stamp Nano Seconds Register .....             | 23-209 |
| Time Stamp Seconds Update Register .....           | 23-210 |
| Time Stamp Nano Seconds Update Register .....      | 23-210 |
| Time Stamp Addend Register .....                   | 23-211 |
| Time Stamp Target Time Seconds Register .....      | 23-212 |
| Time Stamp Target Time Nano Seconds Register ..... | 23-213 |
| Time Stamp High Second Register .....              | 23-214 |

|                                                     |        |
|-----------------------------------------------------|--------|
| Time Stamp Status Register .....                    | 23-215 |
| PPS Control Register .....                          | 23-216 |
| Time Stamp Auxiliary TS Nano Seconds Register ..... | 23-220 |
| Time Stamp Auxiliary TM Seconds Register .....      | 23-220 |
| Time Stamp PPS Interval Register .....              | 23-221 |
| PPS Width Register .....                            | 23-221 |
| DMA Bus Mode Register .....                         | 23-222 |
| DMA Tx Poll Demand Register .....                   | 23-225 |
| DMA Rx Poll Demand register .....                   | 23-226 |
| DMA Rx Descriptor List Address Register .....       | 23-227 |
| DMA Tx Descriptor List Address Register .....       | 23-228 |
| DMA Status Register .....                           | 23-229 |
| DMA Operation Mode Register .....                   | 23-234 |
| DMA Interrupt Enable Register .....                 | 23-240 |
| DMA Missed Frame Register .....                     | 23-242 |
| DMA Rx Interrupt Watch Dog Register .....           | 23-243 |
| DMA SCB Bus Mode Register .....                     | 23-244 |
| DMA SCB Status Register .....                       | 23-246 |
| DMA Tx Descriptor Current Register .....            | 23-247 |
| DMA Rx Descriptor Current Register .....            | 23-247 |
| DMA Tx Buffer Current Register .....                | 23-248 |
| DMA Rx Buffer Current Register .....                | 23-249 |

## Removable Storage Interface (RSI)

|                                     |      |
|-------------------------------------|------|
| RSI Features .....                  | 24-1 |
| RSI Functional Description .....    | 24-2 |
| RSI Register List .....             | 24-2 |
| ADSP-BF60x RSI Interrupt List ..... | 24-4 |
| ADSP-BF60x RSI Trigger List .....   | 24-4 |
| RSI Clock Configuration .....       | 24-5 |
| RSI Interface Configuration .....   | 24-5 |

|                                      |       |
|--------------------------------------|-------|
| Card Detection .....                 | 24-6  |
| RSI Power Saving Configuration ..... | 24-8  |
| RSI Command-Response Interface ..... | 24-9  |
| IDLE State .....                     | 24-12 |
| PEND State .....                     | 24-13 |
| SEND State .....                     | 24-13 |
| WAIT State .....                     | 24-13 |
| RECEIVE State .....                  | 24-14 |
| CEATA_INT_WAIT State .....           | 24-14 |
| CEATA_INT_DIS State .....            | 24-14 |
| RSI Command Path CRC .....           | 24-14 |
| RSI Data Interface .....             | 24-15 |
| RSI Data Transmit Path .....         | 24-17 |
| RSI Data Receive Path .....          | 24-18 |
| RSI Data Path CRC .....              | 24-18 |
| RSI Data FIFO .....                  | 24-19 |
| Card Busy/Ready Detection .....      | 24-20 |
| SDIO Support .....                   | 24-20 |
| RSI Block Diagram .....              | 24-21 |
| RSI Signal Descriptions .....        | 24-22 |
| RSI Architectural Concepts .....     | 24-24 |
| RSI Operating Modes .....            | 24-25 |
| Card Identification Mode .....       | 24-25 |
| Data Transfer Mode .....             | 24-25 |
| DMA Data Transfers .....             | 24-26 |
| Core Data Transfers .....            | 24-26 |
| Boot Mode .....                      | 24-26 |
| Normal Boot Mode .....               | 24-27 |
| Alternate Boot Mode .....            | 24-28 |
| Sleep Mode .....                     | 24-29 |

|                                            |       |
|--------------------------------------------|-------|
| RSI Data Transfer Modes .....              | 24-30 |
| RSI Signal Modes .....                     | 24-30 |
| RSI Event Control .....                    | 24-30 |
| RSI Interrupt Signals .....                | 24-30 |
| RSI Status and Error Signals .....         | 24-30 |
| RSI Programming Model .....                | 24-30 |
| Card Identification .....                  | 24-31 |
| SD Card Identification Procedure .....     | 24-31 |
| MMC Card Identification Procedure .....    | 24-32 |
| Data transfer .....                        | 24-32 |
| Single Block Write .....                   | 24-32 |
| Using Core .....                           | 24-33 |
| Using DMA .....                            | 24-34 |
| Single Block Read .....                    | 24-34 |
| Using Core .....                           | 24-35 |
| Using DMA .....                            | 24-36 |
| Multiple Block Write .....                 | 24-37 |
| Using Core .....                           | 24-37 |
| Using DMA .....                            | 24-38 |
| Multiple Block Read .....                  | 24-39 |
| Using the Core .....                       | 24-39 |
| Using DMA .....                            | 24-40 |
| RSI Programming Concepts .....             | 24-41 |
| Disabling CRC check .....                  | 24-42 |
| Data End Interrupt .....                   | 24-42 |
| Miscellaneous Programming Guidelines ..... | 24-42 |
| RSI Programming Examples .....             | 24-42 |
| RSI Register Descriptions .....            | 24-42 |
| Control Register .....                     | 24-44 |
| Argument Register .....                    | 24-46 |

|                                                 |       |
|-------------------------------------------------|-------|
| Command Register .....                          | 24-46 |
| Response Command Register .....                 | 24-48 |
| Response 0 Register .....                       | 24-48 |
| Response 1 Register .....                       | 24-49 |
| Response 2 Register .....                       | 24-50 |
| Response 3 Register .....                       | 24-50 |
| Data Timer Register .....                       | 24-51 |
| Data Length Register .....                      | 24-51 |
| Data Control Register .....                     | 24-52 |
| Data Count Register .....                       | 24-53 |
| Status Register .....                           | 24-54 |
| Status Clear Register .....                     | 24-56 |
| Interrupt 0 Mask Register .....                 | 24-57 |
| Interrupt 1 Mask Register .....                 | 24-58 |
| FIFO Counter Register .....                     | 24-60 |
| This register contains bit to dis CCS gen ..... | 24-61 |
| Boot Timing Counter Register .....              | 24-61 |
| Boot Acknowledge Timeout Register .....         | 24-62 |
| Sleep Wakeup Timeout Register .....             | 24-63 |
| Block Size Register .....                       | 24-63 |
| Data FIFO Register .....                        | 24-64 |
| Exception Status Register .....                 | 24-64 |
| Exception Mask Register .....                   | 24-67 |
| Configuration Register .....                    | 24-69 |
| Read Wait Enable Register .....                 | 24-72 |
| Peripheral Identification Register .....        | 24-72 |
| Peripheral Identification Register .....        | 24-73 |
| Peripheral Identification Register .....        | 24-74 |
| Peripheral Identification Register .....        | 24-74 |

## Serial Peripheral Interface (SPI)

|                                                   |       |
|---------------------------------------------------|-------|
| SPI Features .....                                | 25-1  |
| SPI Functional Description .....                  | 25-1  |
| SPI Register List .....                           | 25-2  |
| ADSP-BF60x SPI Interrupt List .....               | 25-3  |
| ADSP-BF60x SPI Trigger List .....                 | 25-4  |
| SPI Block Diagram .....                           | 25-5  |
| SPI Transfer Protocol .....                       | 25-5  |
| SPI Clock Considerations .....                    | 25-7  |
| Controlling Delay Between Frames .....            | 25-7  |
| SPI Flow Control .....                            | 25-9  |
| SPI Slave Select Operation .....                  | 25-10 |
| Beginning and Ending a Non-DMA SPI Transfer ..... | 25-11 |
| SPI Transmit Operation in Non-DMA Mode .....      | 25-12 |
| SPI Receive Operation in Non-DMA Mode .....       | 25-12 |
| SPI Dual I/O Mode .....                           | 25-13 |
| SPI Quad I/O Mode .....                           | 25-14 |
| SPI Fast Mode .....                               | 25-15 |
| SPI Interrupt Signals .....                       | 25-16 |
| SPI Data Interrupts .....                         | 25-16 |
| SPI Status Interrupts .....                       | 25-17 |
| SPI Error Conditions .....                        | 25-18 |
| SPI Programming Concepts .....                    | 25-18 |
| Programming Guidelines .....                      | 25-19 |
| SPI Master Operation in Non-DMA Modes .....       | 25-19 |
| SPI Slave Operation in Non-DMA Modes .....        | 25-20 |
| Configuring SPI DMA Master Mode .....             | 25-20 |
| Configuring SPI DMA Slave Mode Operation .....    | 25-22 |
| SPI Register Descriptions .....                   | 25-23 |
| Control Register .....                            | 25-24 |
| Receive Control Register .....                    | 25-31 |

|                                              |       |
|----------------------------------------------|-------|
| Transmit Control Register .....              | 25-34 |
| Clock Rate Register .....                    | 25-37 |
| Delay Register .....                         | 25-38 |
| Slave Select Register .....                  | 25-39 |
| Received Word Count Register .....           | 25-43 |
| Received Word Count Reload Register .....    | 25-44 |
| Transmitted Word Count Register .....        | 25-45 |
| Transmitted Word Count Reload Register ..... | 25-45 |
| Interrupt Mask Register .....                | 25-46 |
| Interrupt Mask Clear Register .....          | 25-49 |
| Interrupt Mask Set Register .....            | 25-50 |
| Status Register .....                        | 25-51 |
| Masked Interrupt Condition Register .....    | 25-56 |
| Masked Interrupt Clear Register .....        | 25-59 |
| Receive FIFO Data Register .....             | 25-60 |
| Transmit FIFO Data Register .....            | 25-61 |

## Serial Port (SPORT)

|                                       |       |
|---------------------------------------|-------|
| Features .....                        | 26-2  |
| Signal Descriptions .....             | 26-4  |
| Serial Clock .....                    | 26-5  |
| Frame Sync .....                      | 26-6  |
| Data Signals .....                    | 26-7  |
| Transmit Data Valid Signal .....      | 26-7  |
| Functional Description .....          | 26-7  |
| SPORT Register List .....             | 26-8  |
| ADSP-BF60x SPORT Interrupt List ..... | 26-9  |
| ADSP-BF60x SPORT Trigger List .....   | 26-10 |
| Block Diagram .....                   | 26-11 |
| Architectural Concepts .....          | 26-12 |
| Multiplexer Logic .....               | 26-14 |



|                                                                 |       |
|-----------------------------------------------------------------|-------|
| Data Types and Companding .....                                 | 26-16 |
| Companding as a Function .....                                  | 26-17 |
| Transmit Path .....                                             | 26-17 |
| Receive Path .....                                              | 26-18 |
| Sampling Edge .....                                             | 26-19 |
| Premature Frame Sync Error Detection .....                      | 26-20 |
| Support for Edge-Detected and Level-Sensitive Frame Syncs ..... | 26-21 |
| Serial Word Length .....                                        | 26-22 |
| Operating Modes .....                                           | 26-23 |
| Mode Selection .....                                            | 26-25 |
| Standard Serial Mode .....                                      | 26-26 |
| Timing Control Bits .....                                       | 26-26 |
| Clocking Options .....                                          | 26-26 |
| Frame Sync Options .....                                        | 26-27 |
| Data-Dependent Versus Data-Independent Frame Sync .....         | 26-27 |
| Early Versus Late Frame Syncs .....                             | 26-27 |
| Framed Versus Unframed Frame Syncs .....                        | 26-28 |
| Logic Level .....                                               | 26-29 |
| Stereo Modes .....                                              | 26-29 |
| Channel Order First .....                                       | 26-29 |
| I <sup>2</sup> S Mode .....                                     | 26-30 |
| Protocol Configuration Options .....                            | 26-30 |
| Serial Clock and Frame Sync Rates .....                         | 26-31 |
| Left-Justified Mode .....                                       | 26-31 |
| Protocol Configuration Options .....                            | 26-31 |
| Serial Clock and Frame Sync Rates .....                         | 26-32 |
| Right-Justified Mode .....                                      | 26-32 |
| Timing Control Bits .....                                       | 26-34 |
| Serial Clock and Frame Sync Rates .....                         | 26-34 |
| Multichannel Mode .....                                         | 26-34 |

|                                                               |       |
|---------------------------------------------------------------|-------|
| Protocol Configuration Options .....                          | 26-35 |
| Clocking Options .....                                        | 26-36 |
| Frame Sync Options .....                                      | 26-36 |
| Transmit Data Valid (TDV) .....                               | 26-36 |
| Active Channel Selection Registers (SPORT_CS0_A) .....        | 26-37 |
| Multichannel Frame Delay (MFD) .....                          | 26-38 |
| Number of Multichannel Slots (WSIZE) .....                    | 26-38 |
| Window Offset (WOFFSET) .....                                 | 26-38 |
| Companding Selection .....                                    | 26-39 |
| Multichannel DMA Data Packing (MCPDE) .....                   | 26-39 |
| Multichannel Frame .....                                      | 26-39 |
| Packed I2S Mode .....                                         | 26-40 |
| Protocol Configuration Options .....                          | 26-41 |
| Clocking Options .....                                        | 26-41 |
| Frame Sync Options .....                                      | 26-42 |
| Gated Clock Mode .....                                        | 26-42 |
| Data Transfers .....                                          | 26-43 |
| Data Buffers .....                                            | 26-43 |
| Transmit Data Buffers (SPORT_TXPRI_A and SPORT_TXSEC_A) ..... | 26-44 |
| Receive Data Buffers (SPORT_RXPRI_A and SPORT_RXSEC_A) .....  | 26-44 |
| Data Buffer Status .....                                      | 26-44 |
| Data Buffer Packing .....                                     | 26-45 |
| Single Word (Core) Transfers .....                            | 26-45 |
| DMA Transfers .....                                           | 26-46 |
| Error Detection .....                                         | 26-47 |
| Interrupts .....                                              | 26-48 |
| Internal Transfer Completion .....                            | 26-49 |
| Transfer Finish Interrupt (TFI) .....                         | 26-49 |
| SPORT Register Descriptions .....                             | 26-49 |
| Half SPORT 'A' Control Register .....                         | 26-51 |

|                                                           |        |
|-----------------------------------------------------------|--------|
| Half SPORT 'A' Divisor Register .....                     | 26-62  |
| Half SPORT 'A' Multi-channel Control Register .....       | 26-63  |
| Half SPORT 'A' Multi-channel 0-31 Select Register .....   | 26-65  |
| Half SPORT 'A' Multi-channel 32-63 Select Register .....  | 26-66  |
| Half SPORT 'A' Multi-channel 64-95 Select Register .....  | 26-66  |
| Half SPORT 'A' Multi-channel 96-127 Select Register ..... | 26-67  |
| Half SPORT 'A' Error Register .....                       | 26-68  |
| Half SPORT 'A' Multi-channel Status Register .....        | 26-71  |
| Half SPORT 'A' Control 2 Register .....                   | 26-71  |
| Half SPORT 'A' Tx Buffer (Primary) Register .....         | 26-72  |
| Half SPORT 'A' Rx Buffer (Primary) Register .....         | 26-73  |
| Half SPORT 'A' Tx Buffer (Secondary) Register .....       | 26-74  |
| Half SPORT 'A' Rx Buffer (Secondary) Register .....       | 26-75  |
| Half SPORT 'B' Control Register .....                     | 26-76  |
| Half SPORT 'B' Divisor Register .....                     | 26-87  |
| Half SPORT 'B' Multi-channel Control Register .....       | 26-88  |
| Half SPORT 'B' Multi-channel 0-31 Select Register .....   | 26-90  |
| Half SPORT 'B' Multi-channel 32-63 Select Register .....  | 26-91  |
| Half SPORT 'B' Multichannel 64-95 Select Register .....   | 26-91  |
| Half SPORT 'B' Multichannel 96-127 Select Register .....  | 26-92  |
| Half SPORT 'B' Error Register .....                       | 26-93  |
| Half SPORT 'B' Multi-channel Status Register .....        | 26-96  |
| Half SPORT 'B' Control 2 Register .....                   | 26-96  |
| Half SPORT 'B' Tx Buffer (Primary) Register .....         | 26-97  |
| Half SPORT 'B' Rx Buffer (Primary) Register .....         | 26-98  |
| Half SPORT 'B' Tx Buffer (Secondary) Register .....       | 26-99  |
| Half SPORT 'B' Rx Buffer (Secondary) Register .....       | 26-100 |

## ADC Control Module (ACM)

|                                  |      |
|----------------------------------|------|
| ACM Features .....               | 27-2 |
| ACM Functional Description ..... | 27-3 |

|                                                                                    |       |
|------------------------------------------------------------------------------------|-------|
| ACM Event Handling Latency .....                                                   | 27-5  |
| ACM Timing Specifications .....                                                    | 27-6  |
| ACM External Pin Timing .....                                                      | 27-7  |
| Case 1—Chip Select Asserted During the High Phase of ACLK (CLKPOL=0) .....         | 27-8  |
| Case 2—Chip Select Asserted During the Low Phase of ACLK (CLKPOL=0) .....          | 27-9  |
| Case 3—Chip Select Asserted Right Before the Falling Edge of ACLK (CLKPOL=1) ..... | 27-9  |
| Case 4—Chip Select Asserted Right Before the Rising Edge of ACLK (CLKPOL=0) .....  | 27-10 |
| Case 5—ACLK Polarity Set to 1 (CLKPOL=1) .....                                     | 27-10 |
| ACM Architectural Concepts .....                                                   | 27-11 |
| ACM Block Diagram .....                                                            | 27-11 |
| ACM Trigger Inputs .....                                                           | 27-12 |
| ACM Timers .....                                                                   | 27-14 |
| Event Register Pairs .....                                                         | 27-15 |
| Event Comparators Unit .....                                                       | 27-16 |
| Timing Generation Unit .....                                                       | 27-16 |
| Status Flags and Interrupts .....                                                  | 27-17 |
| Event Order Registers .....                                                        | 27-18 |
| ACM Register List .....                                                            | 27-18 |
| ADSP-BF60x ACM Interrupt List .....                                                | 27-19 |
| ACM Programming Concepts .....                                                     | 27-20 |
| Emulation Mode Use Case .....                                                      | 27-21 |
| Single-Shot Sequencing Mode Emulation .....                                        | 27-21 |
| Continuous Sequencing Mode Emulation .....                                         | 27-22 |
| ACM Register Descriptions .....                                                    | 27-24 |
| ACM Control Register .....                                                         | 27-25 |
| ACM Timing Configuration 0 Register .....                                          | 27-27 |
| ACM Timing Configuration 1 Register .....                                          | 27-28 |
| ACM Status Register .....                                                          | 27-30 |
| ACM Event Status Register .....                                                    | 27-31 |
| ACM Completed Event Interrupt Mask Register .....                                  | 27-33 |

|                                                |       |
|------------------------------------------------|-------|
| ACM Missed Event Status Register .....         | 27-35 |
| ACM Missed Event Interrupt Mask Register ..... | 27-37 |
| ACM Eventn Control Register .....              | 27-39 |
| ACM Eventn Time Register .....                 | 27-40 |
| ACM Eventn Order Register .....                | 27-41 |
| ACM Timer 0 Register .....                     | 27-42 |
| ACM Timer 1 Register .....                     | 27-43 |

## Link Port (LP)

|                                       |       |
|---------------------------------------|-------|
| LP Features .....                     | 28-1  |
| LP Functional Description .....       | 28-1  |
| LP Register List .....                | 28-2  |
| ADSP-BF60x LP Interrupt List .....    | 28-2  |
| ADSP-BF60x LP Trigger List .....      | 28-3  |
| LP Block Diagram .....                | 28-3  |
| External Connections .....            | 28-4  |
| Internal Blocks .....                 | 28-5  |
| LP Architectural Concepts .....       | 28-5  |
| Link Port Protocol .....              | 28-5  |
| Link Port Buffers .....               | 28-8  |
| Handshake for LP enable process ..... | 28-10 |
| Clocking .....                        | 28-10 |
| Multi-Processor Connectivity .....    | 28-11 |
| LP Operating Modes .....              | 28-13 |
| LP Data Transfer Modes .....          | 28-13 |
| DMA Data Transfers .....              | 28-13 |
| Core Data Transfers .....             | 28-13 |
| LP Event Control .....                | 28-13 |
| LP Interrupt Signals .....            | 28-14 |
| Enabling Link Port interrupts .....   | 28-14 |
| LP Status and Error Signals .....     | 28-15 |

|                                             |       |
|---------------------------------------------|-------|
| LP Programming Model .....                  | 28-15 |
| Setting Up LP DMA Transmit Operation .....  | 28-16 |
| Setting Up LP DMA Receive Operation .....   | 28-16 |
| Setting Up LP Core Transmit Operation ..... | 28-17 |
| Setting Up LP Core Receive Operation .....  | 28-18 |
| LP Register Descriptions .....              | 28-18 |
| Control Register .....                      | 28-19 |
| Status Register .....                       | 28-20 |
| Clock Divider Value .....                   | 28-23 |
| Transmit Buffer .....                       | 28-23 |
| Receive Buffer .....                        | 28-24 |
| Shadow Input Transmit Buffer .....          | 28-25 |
| Shadow Output Transmit Buffer .....         | 28-25 |

## **Video Subsystem (VID)**

|                                                    |      |
|----------------------------------------------------|------|
| VID Features .....                                 | 29-1 |
| VID Functional Description .....                   | 29-1 |
| VID Register List .....                            | 29-2 |
| VID Block Diagram .....                            | 29-2 |
| VID Architectural Concepts .....                   | 29-3 |
| PIXC and PVP Co-Processing .....                   | 29-3 |
| PIXC and PVP Preprocessing .....                   | 29-4 |
| PIXC Post Processing .....                         | 29-5 |
| Concurrency of Pre-, Co- and Post-Processing ..... | 29-6 |
| VID Status and Error Signals .....                 | 29-6 |
| VID Programming Model .....                        | 29-7 |
| VID Performance .....                              | 29-7 |
| VID Register Descriptions .....                    | 29-7 |
| Video Subsystem Connect Register .....             | 29-8 |

## **Pipelined Vision Processor (PVP)**

|                                              |       |
|----------------------------------------------|-------|
| PVP Features .....                           | 30-1  |
| PVP Functional Description .....             | 30-2  |
| PVP Register List .....                      | 30-2  |
| ADSP-BF60x PVP Interrupt List .....          | 30-8  |
| ADSP-BF60x PVP Trigger List .....            | 30-9  |
| PVP Block Diagram .....                      | 30-10 |
| Input Formatters (IPFx) .....                | 30-11 |
| Odd/Even Outputs .....                       | 30-12 |
| Windowing .....                              | 30-13 |
| Monochrome Data Reception .....              | 30-14 |
| Receiving Unsigned Data .....                | 30-14 |
| Color Extraction .....                       | 30-14 |
| Pipe Mastering .....                         | 30-19 |
| PPI and PVP Interoperability .....           | 30-19 |
| Output Formatters (OPFx) .....               | 30-26 |
| Data Packing .....                           | 30-28 |
| Output FIFOs .....                           | 30-30 |
| Threshold-Histogram-Compression (THCx) ..... | 30-31 |
| Threshold Unit .....                         | 30-33 |
| Histogram Unit .....                         | 30-35 |
| Compression Unit .....                       | 30-35 |
| Windowing .....                              | 30-36 |
| Convolution (CNVx) .....                     | 30-36 |
| Polar Magnitude and Angle Block (PMAx) ..... | 30-40 |
| Arithmetic Control Unit (ACUx) .....         | 30-43 |
| Pixel Edge Classifier (PEC) .....            | 30-46 |
| PEC 1st Derivative Mode (PEC-1) .....        | 30-49 |
| Angle Mapping .....                          | 30-49 |
| Non-Maximum Suppression .....                | 30-49 |
| Hysteresis Thresholding .....                | 30-50 |

|                                              |       |
|----------------------------------------------|-------|
| Output Format .....                          | 30-50 |
| PEC 2nd Derivative Mode (PEC-2) .....        | 30-53 |
| Zero-Crossing Detection .....                | 30-53 |
| Sub-Pixel Interpolation .....                | 30-54 |
| Angle Classification .....                   | 30-55 |
| Output Format .....                          | 30-56 |
| Integral Image Block (IIMx) .....            | 30-57 |
| Integral Row Mode .....                      | 30-57 |
| Summed Area Table (SAT) Mode .....           | 30-57 |
| Rotated Summed Area Table (RSAT) Mode .....  | 30-58 |
| Using SAT and RSAT Maps .....                | 30-59 |
| Data Types .....                             | 30-60 |
| Controlling Bandwidth .....                  | 30-61 |
| Connectivity .....                           | 30-61 |
| Up Down Scaler (UDS) .....                   | 30-62 |
| Architectural Concepts .....                 | 30-64 |
| Operating Modes .....                        | 30-65 |
| Thresholds and Histograms .....              | 30-65 |
| Sobel with 3x3 or 5x5 Matrix Operation ..... | 30-66 |
| Sobel Output Formats .....                   | 30-67 |
| Canny with PEC in 1st-Derivative Mode .....  | 30-68 |
| LoG with PEC in 2nd-Derivative Mode .....    | 30-69 |
| DoG with PEC in 2nd-Derivative Mode .....    | 30-71 |
| Integral of Input Pixels .....               | 30-72 |
| Integral of Binary Edge Map .....            | 30-73 |
| Integral of Variance .....                   | 30-73 |
| Histogram of Gradients (HoG) .....           | 30-74 |
| Event Control .....                          | 30-74 |
| Interrupt Signals .....                      | 30-74 |
| Status and Error Signals .....               | 30-76 |



|                                                                                        |        |
|----------------------------------------------------------------------------------------|--------|
| Finish Commands .....                                                                  | 30-77  |
| Programming Model .....                                                                | 30-77  |
| Configuring Pipe Structure .....                                                       | 30-78  |
| Register Based Camera Pipe Configuration .....                                         | 30-82  |
| Block Configuration Structure .....                                                    | 30-83  |
| Initial Configuration Fetch .....                                                      | 30-85  |
| Descriptor-based Memory Pipe Configuration .....                                       | 30-85  |
| Dynamic Pipe Control .....                                                             | 30-86  |
| Latency and Pipeline Depth .....                                                       | 30-88  |
| Daisy Chain Configuration .....                                                        | 30-89  |
| Job Lists .....                                                                        | 30-90  |
| Configuration Descriptor List (Static Job List) .....                                  | 30-91  |
| Global Trigger .....                                                                   | 30-91  |
| Configuration Descriptor List (Dynamic Job List) .....                                 | 30-92  |
| Control Data Flow (Dynamic Job List, Auto-completion Mode and Back-to-Back Mode) ..... | 30-92  |
| Data Out Flow (Static Job List) .....                                                  | 30-94  |
| Data Out Flow (Dynamic Job List) .....                                                 | 30-94  |
| Data Out Flow (Dynamic Job List, Back-to-back Mode) .....                              | 30-95  |
| Data Out Flow (Dynamic Job List, Auto-Completion Mode) .....                           | 30-96  |
| Data In Descriptor List (Static Job List) .....                                        | 30-97  |
| Data In Flow (Dynamic Job List) .....                                                  | 30-97  |
| Starting a Job List .....                                                              | 30-98  |
| Data Flow (Static Job List, Back-to-Back Mode) .....                                   | 30-98  |
| Data Flow (Static Job List, Auto completion Mode) .....                                | 30-99  |
| Data Flow (Dynamic Job List, Back-to-Back Mode) .....                                  | 30-100 |
| Data Flow (Dynamic Job List, Auto-Completion) .....                                    | 30-101 |
| Status (Histogram) Reports .....                                                       | 30-102 |
| Status Word Counters .....                                                             | 30-103 |
| Block Status Structure .....                                                           | 30-104 |
| PVP Register Descriptions .....                                                        | 30-105 |

|                                          |        |
|------------------------------------------|--------|
| Revision ID .....                        | 30-110 |
| Control .....                            | 30-110 |
| Interrupt Mask n .....                   | 30-112 |
| Status .....                             | 30-116 |
| Interrupt Latch Status n .....           | 30-124 |
| Interrupt Request n .....                | 30-129 |
| OPFn (Camera Pipe) Configuration .....   | 30-133 |
| OPFn (Camera Pipe) Control .....         | 30-135 |
| OPF3 (Memory Pipe) Configuration .....   | 30-137 |
| OPF3 (Memory Pipe) Control .....         | 30-139 |
| PEC Configuration .....                  | 30-141 |
| PEC Control .....                        | 30-142 |
| PEC Lower Hysteresis Threshold .....     | 30-143 |
| PEC Upper Hysteresis Threshold .....     | 30-144 |
| PEC Weak Zero Crossing Threshold .....   | 30-145 |
| PEC Strong Zero Crossing Threshold ..... | 30-145 |
| IIMn Configuration .....                 | 30-146 |
| IIMn Control .....                       | 30-148 |
| IIMn Scaling Values .....                | 30-149 |
| IIMn Signed Overflow Status .....        | 30-150 |
| IIMn Unsigned Overflow Status .....      | 30-151 |
| ACU Configuration .....                  | 30-151 |
| ACU Control .....                        | 30-154 |
| ACU SUM Constant .....                   | 30-157 |
| ACU PROD Constant .....                  | 30-157 |
| ACU Shift Constant .....                 | 30-158 |
| ACU Lower Sat Threshold Min .....        | 30-159 |
| ACU Upper Sat Threshold Max .....        | 30-159 |
| UDS Configuration .....                  | 30-160 |
| UDS Control .....                        | 30-161 |

|                                                  |        |
|--------------------------------------------------|--------|
| UDS Output HCNT .....                            | 30-162 |
| UDS Output VCNT .....                            | 30-162 |
| UDS HAVG .....                                   | 30-163 |
| UDS VAVG .....                                   | 30-163 |
| IPF0 (Camera Pipe) Configuration .....           | 30-164 |
| IPFn (Camera/Memory Pipe) Pipe Control .....     | 30-165 |
| IPFn (Camera/Memory Pipe) Control .....          | 30-166 |
| IPFn (Camera/Memory Pipe) TAG Value .....        | 30-169 |
| IPFn (Camera/Memory Pipe) Frame Count .....      | 30-170 |
| IPFn (Camera/Memory Pipe) Horizontal Count ..... | 30-171 |
| IPFn (Camera/Memory Pipe) Vertical Count .....   | 30-171 |
| IPF0 (Camera Pipe) Horizontal Position .....     | 30-172 |
| IPF0 (Camera Pipe) Vertical Position .....       | 30-172 |
| IPFn (Camera/Memory Pipe) TAG Status .....       | 30-173 |
| IPF1 (Memory Pipe) Configuration .....           | 30-174 |
| CNVn Configuration .....                         | 30-175 |
| CNVn Control .....                               | 30-176 |
| CNVn Coefficients 0,0 and 0,1 .....              | 30-177 |
| CNVn Coefficients 0,2 and 0,3 .....              | 30-178 |
| CNVn Coefficient 0,4 .....                       | 30-178 |
| CNVn Coefficients 1,0 and 1,1 .....              | 30-179 |
| CNVn Coefficients 1,2 and 1,3 .....              | 30-180 |
| CNVn Coefficient 1,4 .....                       | 30-180 |
| CNVn Coefficients 2,0 and 2,1 .....              | 30-181 |
| CNVn Coefficients 2,2 and 2,3 .....              | 30-181 |
| CNVn Coefficient 2,4 .....                       | 30-182 |
| CNVn Coefficients 3,0 and 3,1 .....              | 30-183 |
| CNVn Coefficients 3,2 and 3,3 .....              | 30-183 |
| CNVn Coefficient 3,4 .....                       | 30-184 |
| CNVn Coefficients 4,0 and 4,1 .....              | 30-184 |

|                                          |        |
|------------------------------------------|--------|
| CNVn Coefficients 4,2 and 4,3 .....      | 30-185 |
| CNVn Coefficient 4,4 .....               | 30-186 |
| CNVn Scaling Factor .....                | 30-186 |
| THCn Configuration .....                 | 30-187 |
| THCn Control .....                       | 30-189 |
| THCn Histogram Frame Count .....         | 30-192 |
| THCn Max RLE Reports .....               | 30-193 |
| THCn Min Clip Value .....                | 30-193 |
| THCn Clip Min Threshold .....            | 30-194 |
| THCn Clip Max Threshold .....            | 30-194 |
| THCn Max Clip Value .....                | 30-195 |
| THCn Threshold Value 0 .....             | 30-196 |
| THCn Threshold Value 1 .....             | 30-196 |
| THCn Threshold Value 2 .....             | 30-197 |
| THCn Threshold Value 3 .....             | 30-197 |
| THCn Threshold Value 4 .....             | 30-198 |
| THCn Threshold Value 5 .....             | 30-199 |
| THCn Threshold Value 6 .....             | 30-199 |
| THCn Threshold Value 7 .....             | 30-200 |
| THCn Threshold Value 8 .....             | 30-200 |
| THCn Threshold Value 9 .....             | 30-201 |
| THCn Threshold Value 10 .....            | 30-202 |
| THCn Threshold Value 11 .....            | 30-202 |
| THCn Threshold Value 12 .....            | 30-203 |
| THCn Threshold Value 13 .....            | 30-203 |
| THCn Threshold Value 14 .....            | 30-204 |
| THCn Threshold Value 15 .....            | 30-205 |
| THCn Histogram Horizontal Position ..... | 30-205 |
| THCn Histogram Vertical Position .....   | 30-206 |
| THCn Histogram Horizontal Count .....    | 30-206 |

|                                         |        |
|-----------------------------------------|--------|
| THCn Histogram Vertical Count .....     | 30-207 |
| THCn RLE Horizontal Position .....      | 30-208 |
| THCn RLE Vertical Position .....        | 30-208 |
| THCn RLE Horizontal Count .....         | 30-209 |
| THCn RLE Vertical Count .....           | 30-210 |
| THCn Histogram Frame Count Status ..... | 30-210 |
| THCn Histogram Counter Value 0 .....    | 30-211 |
| THCn Histogram Counter Value 1 .....    | 30-212 |
| THCn Histogram Counter Value 2 .....    | 30-212 |
| THCn Histogram Counter Value 3 .....    | 30-213 |
| THCn Histogram Counter Value 4 .....    | 30-213 |
| THCn Histogram Counter Value 5 .....    | 30-214 |
| THCn Histogram Counter Value 6 .....    | 30-215 |
| THCn Histogram Counter Value 7 .....    | 30-215 |
| THCn Histogram Counter Value 8 .....    | 30-216 |
| THCn Histogram Counter Value 9 .....    | 30-216 |
| THCn Histogram Counter Value 10 .....   | 30-217 |
| THCn Histogram Counter Value 11 .....   | 30-218 |
| THCn Histogram Counter Value 12 .....   | 30-218 |
| THCn Histogram Counter Value 13 .....   | 30-219 |
| THCn Histogram Counter Value 14 .....   | 30-219 |
| THCn Histogram Counter Value 15 .....   | 30-220 |
| THCn Number of RLE Reports .....        | 30-221 |
| PMA Configuration .....                 | 30-221 |

## Enhanced Parallel Peripheral Interface (EPPI)

|                                   |      |
|-----------------------------------|------|
| EPPI Features .....               | 31-1 |
| EPPI Functional Description ..... | 31-2 |
| RGB Data Formats .....            | 31-2 |
| Data Clipping .....               | 31-2 |
| Data Mirroring .....              | 31-3 |

|                                                                    |       |
|--------------------------------------------------------------------|-------|
| Windowing .....                                                    | 31-4  |
| Preamble, Blanking and Stripping Support .....                     | 31-4  |
| EPPI Register List .....                                           | 31-5  |
| ADSP-BF60x EPPI Interrupt List .....                               | 31-6  |
| ADSP-BF60x EPPI Trigger List .....                                 | 31-7  |
| EPPI Definitions .....                                             | 31-8  |
| EPPI Block Diagram .....                                           | 31-9  |
| EPPI Architectural Concepts .....                                  | 31-9  |
| EPPI Interface .....                                               | 31-10 |
| Reset Operation .....                                              | 31-11 |
| Frame Sync Polarity and Sampling Edge .....                        | 31-11 |
| DMA .....                                                          | 31-12 |
| Pixel Pipe Interface (PxP) .....                                   | 31-13 |
| EPPI Clock .....                                                   | 31-16 |
| EPPI Operating Modes .....                                         | 31-17 |
| ITU-R 656 Modes .....                                              | 31-17 |
| ITU-R 656 Background .....                                         | 31-17 |
| ITU-R 656 Input Modes .....                                        | 31-22 |
| Entire Field .....                                                 | 31-23 |
| Active Video .....                                                 | 31-23 |
| Vertical Blanking Interval (VBI) .....                             | 31-23 |
| ITU-R 656 Output in GP Transmit Modes .....                        | 31-24 |
| Frame Synchronization in ITU-R 656 Modes .....                     | 31-25 |
| Ignoring Premature External Frame Syncs for Data Consistency ..... | 31-26 |
| General-Purpose EPPI Modes .....                                   | 31-26 |
| GP 0 FS Mode .....                                                 | 31-27 |
| Frame Synchronization in GP 0 FS External Trigger Mode .....       | 31-27 |
| Frame Synchronization in GP 0 FS Internal Trigger Mode .....       | 31-27 |
| GP 1 FS Mode .....                                                 | 31-27 |
| GP 2 FS Mode .....                                                 | 31-27 |

|                                                                    |       |
|--------------------------------------------------------------------|-------|
| DEN functionality in GP 2 FS Transmit Mode .....                   | 31-28 |
| GP 3 FS Mode .....                                                 | 31-28 |
| Supported Data Formats .....                                       | 31-29 |
| EPPI Receive Data Format .....                                     | 31-29 |
| EPPI Transmit Data Format .....                                    | 31-31 |
| EPPI Data Transfer Modes .....                                     | 31-32 |
| Data Packing for Receive Modes .....                               | 31-32 |
| Data Packing for Transmit Modes .....                              | 31-33 |
| Sign-Extension and Zero-Filling .....                              | 31-33 |
| Split Receive Modes .....                                          | 31-34 |
| Split Transmit Modes .....                                         | 31-34 |
| Clock Gating .....                                                 | 31-34 |
| Support for Delayed Start of EPPI Frame Syncs .....                | 31-35 |
| Ignoring Premature External Frame Syncs for Data Consistency ..... | 31-35 |
| EPPI Event Control .....                                           | 31-36 |
| EPPI Status, Error and Interrupt Signals .....                     | 31-36 |
| PxP Errors .....                                                   | 31-36 |
| Frame and Line Track Errors .....                                  | 31-37 |
| LTERROVR and LTERRUNDR Errors .....                                | 31-37 |
| FTERROVR and FTERRUNDR Errors .....                                | 31-37 |
| Preamble Error Not Corrected Error .....                           | 31-37 |
| EPPI Programming Model .....                                       | 31-37 |
| Receiving ITU-R 656 Frames .....                                   | 31-38 |
| Transmitting ITU-R 656 Frames in GP Transmit Modes .....           | 31-38 |
| Configuring Transfers in GP 0 FS Mode .....                        | 31-39 |
| Configuring Transfers in GP 1 FS Mode .....                        | 31-39 |
| Configuring Transfers in GP 2 FS Mode .....                        | 31-40 |
| Configuring Transfers in GP 3 FS Mode .....                        | 31-41 |
| Configuring the EPPI to Use the Windowing Feature .....            | 31-41 |
| EPPI Mode Configuration .....                                      | 31-42 |

|                                                                               |       |
|-------------------------------------------------------------------------------|-------|
| Configuring 8-Bit Receive Mode .....                                          | 31-42 |
| Configuring 10/12/14-Bit Receive Modes .....                                  | 31-43 |
| Configuring 16-Bit Receive Mode .....                                         | 31-46 |
| Configuring 18-Bit Receive Mode .....                                         | 31-47 |
| Configuring 24-Bit Receive Mode .....                                         | 31-48 |
| Configuring 8-Bit Split Receive Mode .....                                    | 31-49 |
| Configuring 10/12/14/16-Bit Split Receive Mode with SPLTWRD=0 .....           | 31-52 |
| Configuring 16-Bit Split Receive Mode with SPLTWRD=1 .....                    | 31-54 |
| Configuring 8-Bit Transmit Mode .....                                         | 31-55 |
| Configuring 10/12/14-Bit Transmit Modes .....                                 | 31-56 |
| Configuring 16-Bit Transmit Mode .....                                        | 31-57 |
| Configuring 18-Bit Transmit Mode .....                                        | 31-58 |
| Configuring 24-Bit Transmit Mode .....                                        | 31-58 |
| Configuring 8-Bit Split Transmit Mode .....                                   | 31-59 |
| Configuring 10/12/14/16-Bit Transmit Mode with SPLTWRD = 0 .....              | 31-63 |
| Configuring 16-Bit Split Transmit Mode with SPLTWRD = 1 .....                 | 31-65 |
| EPPI Programming Concepts .....                                               | 31-66 |
| EPPI SMPTE Modes Programming .....                                            | 31-66 |
| EPPI Register Descriptions .....                                              | 31-67 |
| Status Register .....                                                         | 31-68 |
| Horizontal Transfer Count Register .....                                      | 31-71 |
| Horizontal Delay Count Register .....                                         | 31-72 |
| Vertical Transfer Count Register .....                                        | 31-73 |
| Vertical Delay Count Register .....                                           | 31-73 |
| Lines Per Frame Register .....                                                | 31-74 |
| Samples Per Line Register .....                                               | 31-75 |
| Clock Divide Register .....                                                   | 31-75 |
| Control Register .....                                                        | 31-76 |
| FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register ..... | 31-87 |
| FS1 Period Register / EPPI Active Samples Per Line Register .....             | 31-88 |



|                                                                     |       |
|---------------------------------------------------------------------|-------|
| FS2 Width Register / EPPI Lines Of Vertical Blanking Register ..... | 31-89 |
| FS2 Period Register / EPPI Active Lines Per Field Register .....    | 31-90 |
| Interrupt Mask Register .....                                       | 31-90 |
| Clipping Register for ODD (Chroma) Data .....                       | 31-92 |
| Clipping Register for EVEN (Luma) Data .....                        | 31-93 |
| Frame Sync 1 Delay Value .....                                      | 31-94 |
| Frame Sync 2 Delay Value .....                                      | 31-95 |
| Control Register 2 .....                                            | 31-95 |

## Pixel Compositor (PIXC)

|                                               |       |
|-----------------------------------------------|-------|
| PIXC Features .....                           | 32-2  |
| PIXC Functional Description .....             | 32-2  |
| Color Space Converter Output Thresholds ..... | 32-2  |
| YUV Conversion Modes .....                    | 32-2  |
| PIXC Actions .....                            | 32-4  |
| Recommendation .....                          | 32-5  |
| Special Use Cases .....                       | 32-5  |
| Example 1 - Currently Defined Mode .....      | 32-5  |
| Example 1 - Special Usage of this Mode .....  | 32-5  |
| Example 2 - Currently Defined Mode .....      | 32-6  |
| Example 2 - Special Usage of this Mode .....  | 32-6  |
| Example 3 - Currently Defined Mode .....      | 32-7  |
| Example 3 - Special Usage of this Mode .....  | 32-7  |
| Example 4 - Currently Defined Mode .....      | 32-7  |
| Example 4 - Special Usage of this Mode .....  | 32-7  |
| PIXC Register List .....                      | 32-8  |
| ADSP-BF60x PIXC Interrupt List .....          | 32-9  |
| ADSP-BF60x PIXC Trigger List .....            | 32-10 |
| PIXC Definitions .....                        | 32-10 |
| PIXC Block Diagram .....                      | 32-11 |
| PIXC Architectural Concepts .....             | 32-12 |

|                                                             |       |
|-------------------------------------------------------------|-------|
| Pixel Pipe (PxP) Interface .....                            | 32-12 |
| DMA Interface .....                                         | 32-13 |
| PIXC Data Overlay .....                                     | 32-13 |
| PIXC Transparency Control .....                             | 32-16 |
| PIXC Transparency Color .....                               | 32-18 |
| PIXC Color Space Conversion .....                           | 32-19 |
| Case 1 - Image and Overlay in the Same Format .....         | 32-19 |
| Case 2 - Image and Overlay in Different Formats .....       | 32-20 |
| Case 3 - Color Space Conversion Only .....                  | 32-20 |
| PIXC Operating Modes .....                                  | 32-21 |
| PIXC Mode Case 1 - Image/Overlay in the Same Format .....   | 32-21 |
| PIXC Mode Case 2 - Image/Overlay in Different Formats ..... | 32-22 |
| PIXC Mode Case 3 - Color Space Conversion Only .....        | 32-23 |
| PIXC Image/Overlay/Format Actions .....                     | 32-24 |
| PIXC Image/Overlay/Format Recommendations .....             | 32-25 |
| PIXC Image/Overlay/Format Special Use Cases .....           | 32-25 |
| Example 1 - YUV 4:2:2 to YUV 4:4:4 or LCD/RGB .....         | 32-25 |
| Example 2 - YUV 4:4:4 to YUV 4:4:4 or YUV 4:2:2 .....       | 32-26 |
| Example 3 - YUV 4:2:2/4:4:4 to YUV 4:4:4 or YUV 4:2:2 ..... | 32-26 |
| Example 4 - YUV 4:4:4 to YUV 4:2:2 .....                    | 32-27 |
| Color Space Conversion Matrix Equations .....               | 32-28 |
| Color Space Converter Output Thresholds .....               | 32-29 |
| YUV Re-Sampling .....                                       | 32-29 |
| Up Sampling .....                                           | 32-29 |
| Down Sampling .....                                         | 32-30 |
| PIXC Supported Data Formats .....                           | 32-30 |
| Operation in YUV 4:2:2 Format .....                         | 32-30 |
| Operation in RGB888 Format .....                            | 32-31 |
| Operation in RGB565 Format .....                            | 32-32 |
| Operation in RGB666 Format .....                            | 32-32 |

|                                                |       |
|------------------------------------------------|-------|
| Operation with RGB656 and RGB666 Formats ..... | 32-33 |
| PIXC Interrupt Signals .....                   | 32-33 |
| Trigger List for PIXC module .....             | 32-34 |
| PIXC Programming Model .....                   | 32-34 |
| PIXC Mode Configuration .....                  | 32-35 |
| Performing Data Overlay .....                  | 32-35 |
| Performing Color Space Conversion Only .....   | 32-35 |
| PIXC Register Descriptions .....               | 32-36 |
| Control Register .....                         | 32-37 |
| Pixels Per Line Register .....                 | 32-39 |
| Line Per Frame Register .....                  | 32-40 |
| Overlay A Horizontal Start Register .....      | 32-40 |
| Overlay A Horizontal End Register .....        | 32-41 |
| Overlay A Vertical Start Register .....        | 32-41 |
| Overlay A Vertical End Register .....          | 32-42 |
| Overlay A Transparency Ratio Register .....    | 32-42 |
| Overlay B Horizontal Start Register .....      | 32-43 |
| Overlay B Horizontal End Register .....        | 32-43 |
| Overlay B Vertical Start Register .....        | 32-44 |
| Overlay B Vertical End Register .....          | 32-44 |
| Overlay B Transparency Ratio Register .....    | 32-45 |
| Interrupt Status Register .....                | 32-45 |
| RY Conversion Component Register .....         | 32-46 |
| GU Conversion Component Register .....         | 32-47 |
| BV Conversion Component Register .....         | 32-48 |
| Conversion Bias Register .....                 | 32-49 |
| Transparency Color Register .....              | 32-49 |
| Revision Id .....                              | 32-50 |

## Reset Control Unit (RCU)

|                         |      |
|-------------------------|------|
| RCU Register List ..... | 33-1 |
|-------------------------|------|

|                                         |       |
|-----------------------------------------|-------|
| ADSP-BF60x RCU Trigger List .....       | 33-2  |
| RCU Definitions .....                   | 33-2  |
| RCU Architectural Concepts .....        | 33-3  |
| Reset Sources .....                     | 33-3  |
| RCU Status and Error Signals .....      | 33-4  |
| Resetting a Core .....                  | 33-4  |
| BF60x Specific Information .....        | 33-5  |
| RCU Register Descriptions .....         | 33-5  |
| Control Register .....                  | 33-6  |
| Status Register .....                   | 33-7  |
| Core Reset Control Register .....       | 33-9  |
| Core Reset Status Register .....        | 33-10 |
| System Interface Disable Register ..... | 33-11 |
| System Interface Status Register .....  | 33-12 |
| SVECT Lock Register .....               | 33-13 |
| Boot Code Register .....                | 33-14 |
| Software Vector Register n .....        | 33-15 |

## Booting the Processor (Boot)

|                          |      |
|--------------------------|------|
| Boot Loader Stream ..... | 34-1 |
| Block Structure .....    | 34-3 |
| Block Code .....         | 34-4 |
| TARGET_ADDRESS .....     | 34-5 |
| BYTE_COUNT .....         | 34-5 |
| ARGUMENT .....           | 34-5 |
| Block Types .....        | 34-5 |
| Normal Block .....       | 34-6 |
| First Block .....        | 34-6 |
| Final Block .....        | 34-6 |
| Indirect Block .....     | 34-7 |
| Ignore Block .....       | 34-8 |

|                                                              |       |
|--------------------------------------------------------------|-------|
| Init Block .....                                             | 34-8  |
| Callback Block .....                                         | 34-9  |
| Callback Block Used in Conjunction with Indirect Block ..... | 34-10 |
| Quick Boot Block .....                                       | 34-11 |
| Save Block .....                                             | 34-11 |
| Forward Block .....                                          | 34-11 |
| Forwarding to SPI .....                                      | 34-12 |
| Forwarding to the Link Ports .....                           | 34-13 |
| Multi-DXE Boot Streams .....                                 | 34-14 |
| Single-Block Boot Streams .....                              | 34-16 |
| Direct Code Execution .....                                  | 34-17 |
| Boot Modes .....                                             | 34-18 |
| No-Boot Mode .....                                           | 34-19 |
| Memory Boot Mode .....                                       | 34-19 |
| Padding Memory .....                                         | 34-21 |
| Auto Detection .....                                         | 34-21 |
| Default Static Memory Controller (SMC) settings .....        | 34-22 |
| Run-time API .....                                           | 34-23 |
| RSI Master Boot Mode .....                                   | 34-24 |
| PageMode Customization .....                                 | 34-24 |
| Callback Customization .....                                 | 34-24 |
| RSI Code .....                                               | 34-24 |
| Run-Time API .....                                           | 34-25 |
| Notes on eMMC .....                                          | 34-27 |
| SPI Master Boot Mode .....                                   | 34-27 |
| SPI Device Detection Routine .....                           | 34-28 |
| Run-time API .....                                           | 34-30 |
| SPI Slave Boot Mode .....                                    | 34-31 |
| Run-Time API .....                                           | 34-35 |
| Link Port Slave Boot Mode .....                              | 34-35 |

|                                     |       |
|-------------------------------------|-------|
| Run-time API .....                  | 34-36 |
| UART Slave Boot Mode .....          | 34-37 |
| Autobaud Detection .....            | 34-38 |
| Run-time API .....                  | 34-39 |
| Boot Programming Model .....        | 34-40 |
| Load Functions .....                | 34-40 |
| Page Mode .....                     | 34-41 |
| Changing Settings at Run Time ..... | 34-41 |
| CRC32 Protection .....              | 34-42 |
| Error Handler .....                 | 34-42 |
| Fault Management .....              | 34-43 |
| Callable API Overview .....         | 34-44 |
| System Control .....                | 34-44 |
| Functional Description .....        | 34-44 |
| Boot Kernel .....                   | 34-47 |
| Boot Kernel .....                   | 34-47 |
| Boot Routine .....                  | 34-47 |
| CRC 32 Polynomial .....             | 34-48 |
| CRC Initcode .....                  | 34-48 |
| ECC Protection .....                | 34-49 |
| Execute .....                       | 34-49 |
| Forward Config .....                | 34-51 |
| Get Address .....                   | 34-51 |
| Functional Description .....        | 34-52 |
| Mem Compare .....                   | 34-52 |
| Memory Copy .....                   | 34-53 |
| Memory CRC .....                    | 34-53 |
| Memory Fill .....                   | 34-54 |
| Software Built-in Self Test .....   | 34-54 |
| Bootling Data Structures .....      | 34-55 |

|                                                  |       |
|--------------------------------------------------|-------|
| STRUCT_ROM_SYSCTRL .....                         | 34-56 |
| STRUCT_ROM_BOOT_BUFFER .....                     | 34-56 |
| STRUCT_ROM_BOOT_CONFIG .....                     | 34-57 |
| STRUCT_ROM_BOOT_HEADER .....                     | 34-59 |
| STRUCT_ROM_BOOT_SPI .....                        | 34-59 |
| Wakeup From Hibernate .....                      | 34-59 |
| CGU Initialization after Wakeup .....            | 34-60 |
| DDR Controller Initialization after Wakeup ..... | 34-60 |
| BFLAG_WAKEUP and BFLAG_QUICKBOOT .....           | 34-61 |
| DPM Register List .....                          | 34-61 |
| DPM Restore .....                                | 34-63 |
| Reset and Power-up .....                         | 34-64 |
| Reset Vector .....                               | 34-64 |
| Servicing Reset Interrupts .....                 | 34-65 |
| NMI and RESOUT .....                             | 34-65 |
| Memory Initialization .....                      | 34-65 |
| Debugging .....                                  | 34-65 |
| Boot ROM Revision Control .....                  | 34-67 |
| Boot ROM Revision Control .....                  | 34-67 |
| Boot Register Reference .....                    | 34-68 |
| Status Register .....                            | 34-68 |
| Software Vector Register n .....                 | 34-71 |
| Boot Code Register .....                         | 34-72 |

## System Debug Unit (SDU)

|                                     |      |
|-------------------------------------|------|
| SDU Features .....                  | 35-1 |
| SDU Functional Description .....    | 35-1 |
| SDU Register List .....             | 35-2 |
| ADSP-BF60x SDU Interrupt List ..... | 35-2 |
| ADSP-BF60x SDU Trigger List .....   | 35-3 |
| SDU Definitions .....               | 35-3 |

|                                               |       |
|-----------------------------------------------|-------|
| SDU Block Diagram .....                       | 35-3  |
| JTAG Tap Controller (JTC) Block Diagram ..... | 35-4  |
| JTC Core Emulation .....                      | 35-5  |
| JTC Instruction Register (IR) .....           | 35-6  |
| Memory Access Controller (MAC) .....          | 35-7  |
| MAC Direct Access .....                       | 35-7  |
| MAC DMA Access .....                          | 35-8  |
| Group Halt .....                              | 35-8  |
| Group Halt Status .....                       | 35-9  |
| Group Halt Masters .....                      | 35-9  |
| Group Halt Slaves .....                       | 35-9  |
| SDU Programming Concepts .....                | 35-10 |
| Core Control .....                            | 35-10 |
| Memory and Register Access .....              | 35-10 |
| Background Telemetry Channel (BTC) .....      | 35-10 |
| Statistical Profiling .....                   | 35-11 |
| Power Management Support .....                | 35-11 |
| System Reset Support .....                    | 35-11 |
| SDU Register Descriptions .....               | 35-11 |
| ID Code Register .....                        | 35-12 |
| Control Register .....                        | 35-13 |
| Status Register .....                         | 35-15 |
| Memory Access Control Register .....          | 35-21 |
| Memory Access Address Register .....          | 35-22 |
| Memory Access Data Register .....             | 35-23 |
| DMA Read Data Register .....                  | 35-24 |
| DMA Write Data Register .....                 | 35-24 |
| Message Register .....                        | 35-25 |
| Message Set Register .....                    | 35-26 |
| Message Clear Register .....                  | 35-27 |



|                           |       |
|---------------------------|-------|
| Group Halt Register ..... | 35-27 |
|---------------------------|-------|

## System Watchpoint Unit (SWU)

|                                               |       |
|-----------------------------------------------|-------|
| SWU Features .....                            | 36-1  |
| SWU Functional Description .....              | 36-1  |
| Match Block .....                             | 36-1  |
| SWU Register List .....                       | 36-1  |
| ADSP-BF60x SWU Interrupt List .....           | 36-2  |
| ADSP-BF60x SWU Trigger List .....             | 36-3  |
| SWU Definitions .....                         | 36-4  |
| SWU Architectural Concepts .....              | 36-4  |
| SWU Flow Diagram .....                        | 36-4  |
| SCB Interface .....                           | 36-5  |
| SWU Block Diagram .....                       | 36-5  |
| SWU Event Control .....                       | 36-6  |
| SWU Interrupts .....                          | 36-6  |
| SWU Status and Errors .....                   | 36-6  |
| SWU Operating Modes .....                     | 36-7  |
| Bandwidth Mode .....                          | 36-7  |
| Watchpoint Mode .....                         | 36-7  |
| SWU Programming Model .....                   | 36-7  |
| SWU Programming Concepts .....                | 36-7  |
| SWU Mode Configuration .....                  | 36-8  |
| Configuring the SWU for Bandwidth Mode .....  | 36-8  |
| Configuring the SWU for Watchpoint Mode ..... | 36-9  |
| SWU Register Descriptions .....               | 36-9  |
| Global Control Register .....                 | 36-10 |
| Global Status Register .....                  | 36-11 |
| Control Register n .....                      | 36-15 |
| Lower Address Register n .....                | 36-21 |
| Upper Address Register n .....                | 36-22 |

ID Register n ..... 36-22

Count Register n ..... 36-23

Target Register n ..... 36-24

Bandwidth History Register n ..... 36-25

Current Register n ..... 36-25

**Index**

# Preface

Thank you for purchasing and developing systems using an enhanced Blackfin processor from Analog Devices.

## Purpose of This Manual

The *ADSP-BF60x Blackfin Processor Hardware Reference* provides architectural information about the ADSP-BF60x processors. This hardware reference provides the main architectural information about these processors. The architectural descriptions cover functional blocks, buses, and ports, including all features and processes that they support. For programming information, see the *Blackfin Processor Programming Reference*. For timing, electrical, and package specifications, see the *ADSP-BF60x Blackfin Processor Data Sheet*.

## Intended Audience

The primary audience for this manual is a programmer who is familiar with Analog Devices processors. The manual assumes the audience has a working knowledge of the appropriate processor architecture and instruction set. Programmers who are unfamiliar with Analog Devices processors can use this manual, but should supplement it with other texts, such as hardware reference and programming reference books, that describe their target architecture.

## Manual Contents

This manual consists of the following chapters:

- Preface
- Introduction — Provides a high level overview of the processor, including peripherals, power management, and development tools.
- System Cross Bar (SCB) — Describes on-chip buses, including how data moves through the system.
- Clock Generation Unit (CGU) — Describes the phase locked loop (PLL), PLL control unit (PCU), and CGU, which generate on-chip clocks.
- System Protection Unit (SPU) — Describes the system protection and how the SPU protects system resources from errant writes.

- Dynamic Power Management (DPM) — Describes the clocking, including the PLL, and the dynamic power management controller.
- Core Timers (TMR) — Each processor core has its own dedicated timer. The core timer is clocked by the internal processor clock and is typically used as a system tick clock for generating periodic operating system interrupts.
- System Event Controller (SEC) — Describes the system peripheral interrupts, including setup and clearing of interrupt requests.
- Trigger Routing Unit (TRU) — Describes TRU operations providing system-level sequence control without core intervention.
- Static Memory Controller (SMC) — Describes the static (SRAM) memory controller of the processor and the asynchronous memory interface.
- L2 Memory Controller (L2CTL) — Describes the memory controller and its operation.
- Dynamic Memory Controller (DMC) — Describes the dynamic (DDR2) memory controller of the processor, related registers, configuration, and commands.
- Cyclic Redundancy Check (CRC) — Describes the CRC operations on blocks of data presented to the peripheral.
- DMA Channel (DMA) — Describes the peripheral DMA and Memory DMA controllers, including topics such as performance, software management of DMA, and DMA errors.
- General-Purpose Ports (PORTS) — Describes the general-purpose I/O ports, including the structure of each port, multiplexing, configuring the pins, and generating interrupts.
- Pin Interrupts (PINT) — Describes operation of port pins on the processor, which can request interrupts in either an edge-sensitive or a level-sensitive manner with programmable polarity.
- General-Purpose Timer (TIMER) — Describes the general-purpose timers that can be configured in any of three modes; the core timer that can generate periodic interrupts for a variety of timing functions; and the watchdog timer that can implement software watchdog functions, such as generating events to the Blackfin processor core.
- Watchdog Timer (WDOG) — Describes the watchdog timer.
- General-Purpose Counter (CNT) — Describes the Rotary (up/down) Counter. This counter provides support for manually controlled rotary controllers, such as the volume wheel on a radio device. This unit also supports industrial or motor-control type of wheels.
- Pulsewidth Modulator (PWM) — Describes the The PWM controller—a flexible, programmable, three-phase PWM waveform generator that can be programmed to generate the required switching patterns to drive a three-phase voltage source inverter for ac induction motor (ACIM) or permanent magnet synchronous motor (PMSM) control.

- Universal Async Rx/Tx (UART) — Describes the Universal Asynchronous Receiver/Transmitter port that converts data between serial and parallel formats. The UART supports the half-duplex IrDA SIR protocol as a mode-enabled feature.
- Two Wire Interface (TWI) — Describes the Two Wire Interface (TWI) controller, which allows a device to interface to an Inter IC bus as specified by the *Philips I<sup>2</sup>C Bus Specification version 2.1* dated January 2000.
- Controller Area Network (CAN) — Describes the CAN module, a low bit rate serial interface intended for use in applications where bit rates are typically up to 1Mbit/s.
- Universal Serial Bus (USB) — Describes the USB OTG interface of the processor. This interface provides a low-cost connectivity solution for consumer mobile devices such as cell phones, digital still cameras and MP3 players, allowing these devices to transfer data via a point-to-point USB connection without the need for a PC host.
- 10/100 Ethernet MAC (EMAC) — Describes the Ethernet Media Access Controller (MAC) peripheral that provides a 10/100M bit/s Ethernet interface, compliant to IEEE Std. 802.3-2002, between an MII (Media Independent Interface) and the Blackfin peripheral subsystem. Also, describes the IEEE 1588 engine module and the module's operation.
- Removable Storage Interface (RSI) — Describes the RSI interface for multimedia cards (MMC), secure digital memory cards (SD), secure digital input/output cards (SDIO) and consumer electronic ATA devices (CE-ATA).
- Serial Peripheral Interface (SPI) — Describes the Serial Peripheral Interface (SPI) port that provides an I/O interface to a variety of SPI compatible peripheral devices.
- Serial Port (SPORT) — Describes the independent, synchronous Serial Port Controller which provides an I/O interface to a variety of serial peripheral devices.
- ADC Control Module (ACM) — Describes the ADC control module (ACM), which provides an interface to synchronize the controls between the processor and an analog-to-digital converter (ADC) module.
- Link Port (LP) — Describes the two bidirectional 8-bit wide link ports, which can connect to other processor or peripheral link ports.
- Video Subsystem (VID) — Describes the subsystem's connectivity matrix that interconnects the video related peripherals.
- Pipelined Vision Processor (PVP) — Describes the PVP's support for co-processing and pre-processing of monochrome video frames in ADAS applications, robotic systems, and other machine applications.
- Parallel Peripheral Interface (PPI) — Describes the Parallel Peripheral Interface (PPI) of the processor. The PPI is a half-duplex, bidirectional port accommodating up to 16 bits of data and is used for digital video and data converter applications.

- Pixel Compositor (PIXC) — Describes the PIXC, which provides data overlay, transparent color, and color space conversion support for active (TFT) flat-panel digital color/monochrome LCD displays or analog NTSC/PAL video output.
- Reset Control Unit (RCU)
- Booting — Describes the booting methods, booting process and specific boot modes for the processor.
- System Design — Describes how to use the processor as part of an overall system. It includes information about bus timing and latency numbers, semaphores, and a discussion of the treatment of unused pins.
- Test Features — Describes test features for the processor, discusses the JTAG standard, boundary-scan architecture, instruction and boundary registers, and public instructions.
  - System Debug Unit (SDU)
  - System Watchpoint Unit (SWU)

## What's New in This Manual

This revision (0.4) is a preliminary revision of the *ADSP-BF60x Blackfin Processor Hardware Reference*. This revision adds the register information for the SCB, TMR, DPM, L2CTL, PWM, USB, and CAN chapters. Also, this revision adds supplementary information and corrections throughout the text.

## Technical or Customer Support

You can reach Analog Devices, Inc. Customer Support in the following ways:

- Visit the Embedded Processing and DSP products Web site at [http://www.analog.com/processors/technical\\_support](http://www.analog.com/processors/technical_support)
- E-mail tools questions to <http://mailto:processor.tools.support@analog.com>
- E-mail processor questions to <http://mailto:processor.support@analog.com> (world wide support) or <http://mailto:processor.china@analog.com> (China support)
- Phone questions to 1-800-ANALOGD
- Contact your Analog Devices, Inc. local sales office or authorized distributor
- Send questions by mail to:

Analog Devices, Inc.  
One Technology Way  
P.O. Box 9106  
Norwood, MA 02062-9106 USA

## Registration for MyAnalog.com

[MyAnalog.com](#) is a free feature of the Analog Devices Web site that allows customizing a Web page to display only the latest information about products you are interested in. Click **Register** to use this site. Registration takes about five minutes and serves as a means to select the information you want to receive.

If you are already a registered user, just log on. Your user name is your e-mail address.

## EngineerZone

[EngineerZone](#) is a technical support forum from Analog Devices. It allows you direct access to ADI technical support engineers. You can search FAQs and technical information to get quick answers to your embedded processing and DSP design questions.

Use EngineerZone to connect with other DSP developers who face similar design challenges. You can also use this open forum to share knowledge and collaborate with the ADI support team and your peers. Visit <http://ez.analog.com/welcome> to sign up.

## Social Networking Web Sites

You can now follow Analog Devices news on Twitter and Analog Devices Blackfin products on LinkedIn. To access:

- Twitter: [https://twitter.com/#!/adi\\_news](https://twitter.com/#!/adi_news)
- LinkedIn: <http://www.linkedin.com/groups Analog Devices Blackfin>

## Supported Processors

The following is the list of Analog Devices, Inc. processors supported in CrossCore Embedded Studio® development tools.

### Blackfin (ADSP-BFxxx) Processors

The name *Blackfin* refers to a family of 16-bit, embedded processors. CrossCore Embedded Studio currently supports the following Blackfin families ADSP-BF50x, ADSP-BF51x, ADSP-BF52x, ADSP-BF53x, ADSP-BF54x, ADSP-BF59x, ADSP-BF561, and ADSP-BF60x processors.

### SHARC® (ADSP-21xxx) Processors

The name *SHARC* refers to a family of high-performance, 32-bit, floating-point processors that can be used in speech, sound, graphics, and imaging applications. CrossCore Embedded Studio currently supports the following SHARC families: ADSP-2106x, ADSP-2116x, ADSP-2126x, ADSP-2136x, and ADSP-214xx.

## Product Information

Product information can be obtained from the Analog Devices Web site and CrossCore Embedded Studio online Help system.

## Analog Devices Web Site

The Analog Devices Web site, <http://www.analog.com>, provides information about a broad range of products-analog integrated circuits, amplifiers, converters, and digital signal processors.

To access a complete technical library for each processor family, go to [http://www.analog.com/processors/technical\\_library](http://www.analog.com/processors/technical_library). The manuals selection opens a list of current manuals related to the product as well as a link to the previous revisions of the manuals. When locating your manual title, note a possible errata check mark next to the title that leads to the current correction report against the manual.

Also note, [MyAnalog.com](http://www.analog.com) is a free feature of the Analog Devices Web site that allows customizing a Web page to display only the latest information on products you are interested in. You can also choose to receive weekly e-mail notifications containing updates to the Web pages that meet your interests, including documentation errata against all manuals. [MyAnalog.com](http://www.analog.com) provides access to books, application notes, data sheets, code examples, and more.

Visit [MyAnalog.com](http://www.analog.com) to sign up. If you are already a registered user, just log on. Your user name is your e-mail address.

## CrossCore Embedded Studio Online Documentation

Online documentation comprises the CrossCore Embedded Studio help system, software tools manuals, hardware tools manuals, processor manuals, Dinkum Abridged C++ library, and FLEXnet License Tools software documentation. You can search easily across the entire CrossCore Embedded Studio documentation set for any topic of interest.

For easy printing, supplementary Portable Documentation Format (.pdf) files for all manuals are provided on the CrossCore Embedded Studio product page on the Analog Devices website.

Each documentation file type is described as follows.

| File | Description                                            |
|------|--------------------------------------------------------|
| .chm | Help system files and manuals in Microsoft help format |



| File          | Description                                                                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| .htm or .html | Dinkum Abridged C++ library and FLEXnet License Tools software documentation. Viewing and printing the .html files requires a browser, such as Internet Explorer 6.0 (or higher). |
| .pdf          | CrossCore Embedded Studio and processor manuals in PDF format. Viewing and printing the .pdf files requires a PDF reader, such as Adobe Acrobat Reader (9.0 or higher).           |

## Technical Library CD

The technical library CD contains seminar materials, product highlights, a selection guide, and documentation files of processor manuals, CrossCore Embedded Studio software manuals, and hardware tools manuals for the following processor families: Blackfin, SHARC, TigerSHARC, ADSP-218x, and ADSP-219x.

To order the technical library CD, go to <http://www.analog.com/processors/manuals>, navigate to the manuals page for your processor, click the request CD check mark, and fill out the order form.

Data sheets, which can be downloaded from the Analog Devices Web site, change rapidly, and therefore are not included on the technical library CD. Technical manuals change periodically. Check the Web site for the latest manual revisions and associated documentation errata.

**NOTE:** The technical library CD is no longer available. This media will be replaced with another form of publicly available media in the near future.

## Notation Conventions

Text conventions used in this manual are identified and described as follows. Additional conventions, which apply only to specific chapters, may appear throughout this document.

| Example                   | Description                                                                                                                                                                                   |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Close command (File menu) | Titles in reference sections indicate the location of an item within the CrossCore Embedded Studio IDE's menu system (for example, the <b>Close</b> command appears on the <b>File</b> menu). |

| Example                                      | Description                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| {this   that}                                | Alternative required items in syntax descriptions appear within curly brackets and separated by vertical bars; read the example as <i>this</i> or <i>that</i> . One or the other is required.                                                                                                                                                      |
| [this   that]                                | Optional items in syntax descriptions appear within brackets and separated by vertical bars; read the example as an optional <i>this</i> or <i>that</i> .                                                                                                                                                                                          |
| [this,]                                      | Optional item lists in syntax descriptions appear within brackets delimited by commas and terminated with an ellipse; read the example as an optional comma-separated list of <i>this</i> .                                                                                                                                                        |
| .SECTION                                     | Commands, directives, keywords, and feature names are in text with <i>letter gothic font</i> .                                                                                                                                                                                                                                                     |
| <i>filename</i>                              | Non-keyword placeholders appear in text with italic style format.                                                                                                                                                                                                                                                                                  |
| <b>NOTE:</b> This is an note paragraph.      | <b>Note:</b> For correct operation, ... A <b>Note</b> provides supplementary information on a related topic. In the online version of this book, the word <b>Note</b> appears instead of this symbol.                                                                                                                                              |
| <b>CAUTION:</b> This is a caution paragraph. | <b>Caution:</b> Incorrect device operation may result if ...<br><b>Caution:</b> Device damage may result if ... A <b>Caution</b> identifies conditions or inappropriate usage of the product that could lead to undesirable results or product damage. In the online version of this book, the word <b>Caution</b> appears instead of this symbol. |
| <b>DANGER:</b> This is a danger paragraph.   | <b>Danger:</b> Injury to device users may result if ... A <b>Danger</b> identifies conditions or inappropriate usage of the product that could lead to conditions that are potentially hazardous for the devices users. In the online version of this book, the word <b>Danger</b> appears instead of this symbol.                                 |

# Register Diagram Conventions

Register diagrams use the following conventions:

- The descriptive name of the register appears at the top with the short form of the name.
- If the register is read-only (RO), write-1-to-set (W1S), or write-1-to-clear (W1C), this information appears beside the name.
- If any bits in the register do not follow the overall read/write convention, this is noted in the bit description after the bit name.
- If a bit has a short name, the short name appears first in the bit description, followed by the long name.
- The reset value appears in binary in the individual bits and in hexadecimal to the left of the register.
- Bits marked X have an unknown reset value. Consequently, the reset value of registers that contain such bits is undefined or dependent on pin values at reset.
- Shaded bits are reserved.

**NOTE:** To ensure upward compatibility with future implementations, write back the value that is read for reserved bits in a register, unless otherwise specified.



# 1 Introduction

The ADSP-BF60x processors are members of the Blackfin family of products, incorporating the Analog Devices/Intel Micro Signal Architecture (MSA). Blackfin processors combine a dual-MAC state-of-the-art signal processing engine, the advantages of a clean, orthogonal RISC-like microprocessor instruction set, and single-instruction, multiple-data (SIMD) multimedia capabilities into a single instruction-set architecture.

The processor offers high performance, as well as low static power consumption. Produced with a low-power and low-voltage design methodology, they provide world-class power management and performance.

As shown in the block diagram figure, by integrating a rich set of industry-leading system peripherals and memory, Blackfin processors are the platform of choice for next-generation applications that require RISC-like programmability, multimedia support, and leading-edge signal processing in one integrated package. These applications span a wide array of markets, from automotive systems to embedded industrial, instrumentation and power/motor control applications.

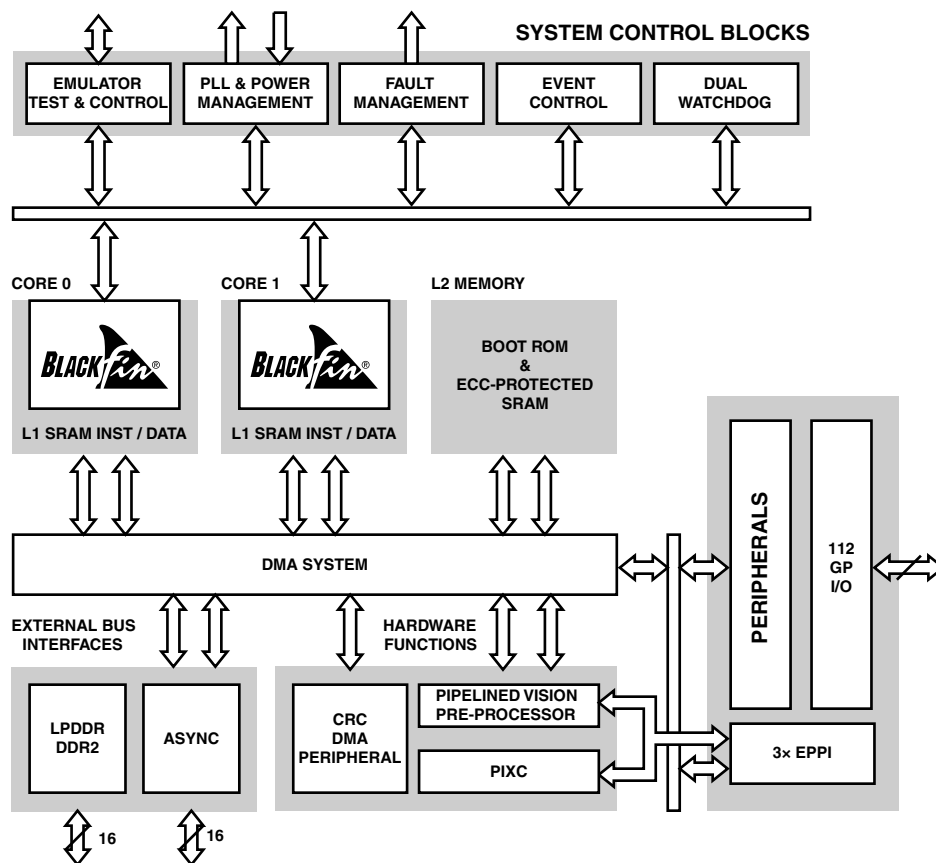


**Figure 1-1:** ADSP-BF60x Functional Block Diagram

## Blackfin Processor Core

As shown in the [ADSP-BF60x Functional Block Diagram](#), the processor integrates two Blackfin processor cores. Each core, shown in [Blackfin Processor Core](#), contains two 16-bit multipliers, two 40-bit accumulators, two 40-bit ALUs, four video ALUs, and a 40-bit shifter. The computation units process 8-, 16-, or 32-bit data from the register file.

The compute register file contains eight 32-bit registers. When performing compute operations on 16-bit operand data, the register file operates as 16 independent 16-bit registers. All operands for compute operations come from the multiported register file and instruction constant fields.

Each MAC can perform a 16-bit by 16-bit multiply in each cycle, accumulating the results into the 40-bit accumulators. Signed and unsigned formats, rounding, and saturation are supported.

The ALUs perform a traditional set of arithmetic and logical operations on 16-bit or 32-bit data. In addition, many special instructions are included to accelerate various signal processing tasks. These include bit operations such as field extract and population count, modulo  $2^{32}$  multiply, divide primitives, saturation and rounding, and sign/exponent detection. The set of video instructions include byte alignment and packing operations, 16-bit and 8-bit adds with clipping, 8-bit average operations, and 8-bit subtract/abso-

lute value/accumulate (SAA) operations. Also provided are the compare/select and vector search instructions.

For certain instructions, two 16-bit ALU operations can be performed simultaneously on register pairs (a 16-bit high half and 16-bit low half of a compute register). If the second ALU is used, quad 16-bit operations are possible.

The 40-bit shifter can perform shifts and rotates and is used to support normalization, field extract, and field deposit instructions.

The program sequencer controls the flow of instruction execution, including instruction alignment and decoding. For program flow control, the sequencer supports PC relative and indirect conditional jumps (with static branch prediction), and subroutine calls. Hardware supports zero-overhead looping. The architecture is fully interlocked, meaning that the programmer need not manage the pipeline when executing instructions with data dependencies.

The address arithmetic unit provides two addresses for simultaneous dual fetches from memory. It contains a multiported register file consisting of four sets of 32-bit index, modify, length, and base registers (for circular buffering), and eight additional 32-bit pointer registers (for C-style indexed stack manipulation).

Blackfin processors support a modified Harvard architecture in combination with a hierarchical memory structure. Level 1 (L1) memories are those that typically operate at the full processor speed with little or no latency. At the L1 level, the instruction memory holds instructions only. The data memory holds data, and a dedicated scratchpad data memory stores stack and local variable information.

In addition, multiple L1 memory blocks are provided, offering a configurable mix of SRAM and cache. The memory management unit (MMU) provides memory protection for individual tasks that may be operating on the core and can protect system registers from unintended access.

The architecture provides three modes of operation: user mode, supervisor mode, and emulation mode. User mode has restricted access to certain system resources, thus providing a protected software environment, while supervisor mode has unrestricted access to the system and core resources.

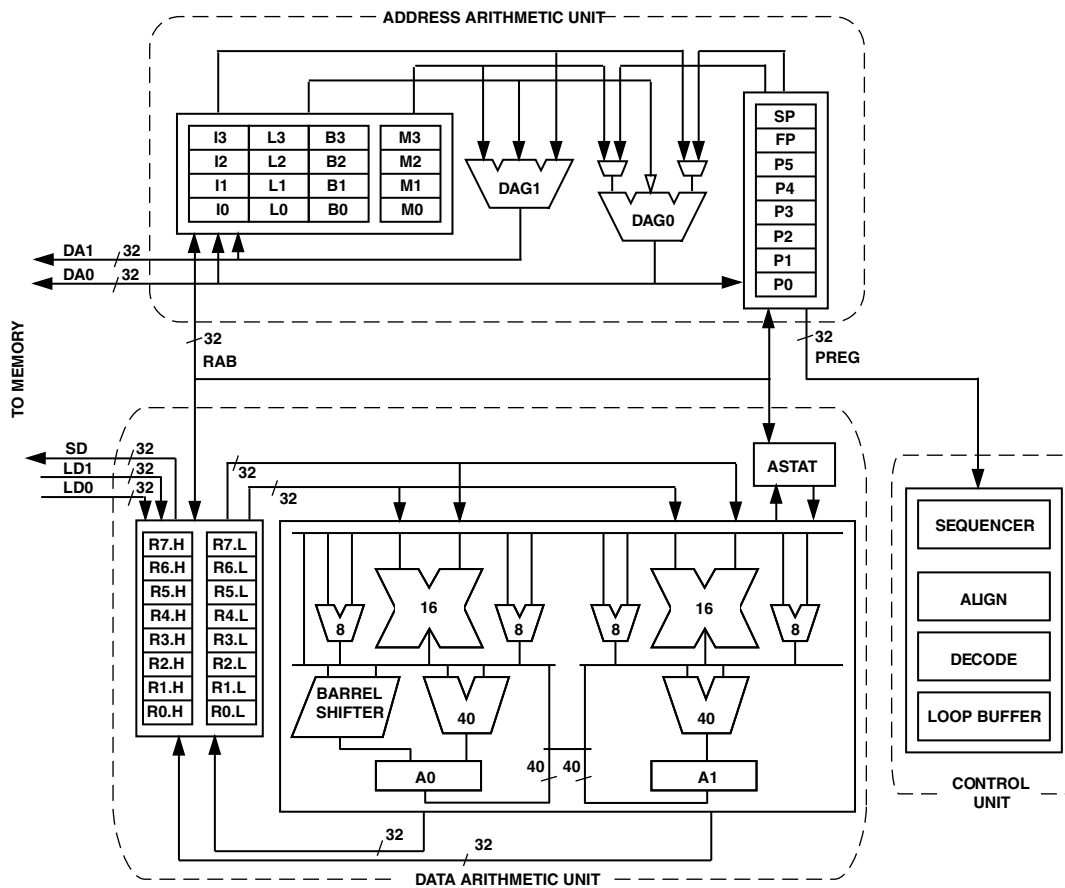


Figure 1-2: Blackfin Processor Core

## Instruction Set Description

The Blackfin processor instruction set has been optimized so that 16-bit opcodes represent the most frequently used instructions, resulting in excellent compiled code density. Complex DSP instructions are encoded into 32-bit opcodes, representing fully featured multifunction instructions. Blackfin processors support a limited multi-issue capability, where a 32-bit instruction can be issued in parallel with two 16-bit instructions, allowing the programmer to use many of the core resources in a single instruction cycle.

The Blackfin processor family assembly language instruction set employs an algebraic syntax designed for ease of coding and readability. The instructions have been specifically tuned to provide a flexible, densely encoded instruction set that compiles to a very small final memory size. The instruction set also provides fully featured multifunction instructions that allow the programmer to use many of the processor core resources in a single instruction. Coupled with many features more often seen on micro controllers, this instruction set is very efficient when compiling C and C++ source code. In addition, the architecture supports both user (algorithm/application code) and supervisor (O/S kernel, device drivers, debuggers, ISRs) modes of operation, allowing multiple levels of access to core processor resources.



The assembly language, which takes advantage of the processor's unique architecture, offers the following advantages:

- Seamlessly integrated DSP/MCU features are optimized for both 8-bit and 16-bit operations.
- A multi-issue load/store modified-Harvard architecture, which supports two 16-bit MAC or four 8-bit ALU + two load/store + two pointer updates per cycle.
- All registers, I/O, and memory are mapped into a unified 4G byte memory space, providing a simplified programming model.
- Control of all asynchronous and synchronous events to the processor is handled by two subsystems: the Core Event Controller (CEC) and the System Event Controller (SEC).
- Micro controller features, such as arbitrary bit and bit-field manipulation, insertion, and extraction; integer operations on 8-, 16-, and 32-bit data-types; and separate user and supervisor stack pointers.
- Code density enhancements, which include intermixing of 16-bit and 32-bit instructions (no mode switching, no code segregation). Frequently used instructions are encoded in 16 bits.

**NOTE:** For more information about the Blackfin instruction set, see the *Blackfin Programming Reference*.

## Processor Safety Features

The ADSP-BF60x processor has been designed for functional safety applications. While the level of safety is mainly dominated by the system concept, the following primitives are provided by the devices to build a robust safety concept.

### Dual Core Supervision

The processor has been implemented as dual-core devices to separate critical tasks to large independency. Software models support mutual supervision of the cores in symmetrical fashion.

### Fault Management

The fault management unit is part of the system event controller (SEC). Any system event, whether a dual-bit uncorrectable ECC error, or any peripheral status interrupt, can be defined as being a "fault". Additionally, the system events can be defined as an interrupt to the cores. If defined as such, the SEC forwards the event to the fault management unit which may automatically reset the entire device for reboot, or simply toggle the SYS\_FAULT output pins to signal off-chip hardware. Optionally, the fault management unit can delay the action taken via a keyed sequence, to provide a final chance for the Blackfin cores to resolve the crisis and to prevent the fault action from being taken.

## System Protection

All system resources and L2 memory banks can be controlled by either the processor cores, memory-to-memory DMA, or the system debug unit (SDU). A system protection unit (SPU) enables write accesses to specific resources that are locked to any of four masters: Core 0, Core 1, Memory DMA, and the System Debug Unit. System protection is enabled in greater granularity for some modules (L2, SEC and GPIO controllers) through a *global lock* concept.

## Bandwidth Monitor

All DMA channels that operate in memory-to-memory mode (Memory DMA, PVP Memory Pipe DMA, PIXC DMA) are equipped with a bandwidth monitor mechanism. They can signal a system event or fault when transactions tend to starve because system buses are fully loaded with higher-priority traffic.

## Private (to each Core) Memory

The L1 memory system is the highest-performance memory available to the Blackfin processor cores.

Each core has its own private L1 memory. The modified Harvard architecture supports two concurrent 32-bit data accesses along with an instruction fetch at full processor speed which provides high bandwidth processor performance. Two separate 64K-byte of data memory blocks partner with an 80K-byte memory block for instruction storage. Each block is multi banked for efficient data exchange through DMA and can be configured as SRAM. Alternatively, 16K bytes of each block can be configured in L1 cache mode. The four-way set-associative instruction cache and the 2 two-way set-associative data caches greatly accelerate memory access performance, especially when accessing external memories.

The L1 memory domain also features a 4K-byte scratchpad SRAM block which is ideal for storing local variables and the software stack. All L1 memory is protected by a multi-parity bit concept, regardless of whether the memory is operating in SRAM or cache mode.

## Shared (by both Cores) Memory

Outside of the L1 domain, L2 and L3 memories are arranged using a Von Neumann topology. The L2 memory domain is a unified instruction and data memory and can hold any mixture of code and data required by the system design. The L2 memory domain is accessible by both Blackfin cores through a dedicated 64-bit interface. It operates at half the frequency of the cores. The processor features 256K bytes of L2 SRAM which is ECC-protected and organized in eight banks. Individual banks can be made private to any of the cores or the DMA subsystem. There is also a 32K-byte single-bank ROM in the L2 domain. It contains boot code and safety functions.

## I/O Memory Space

The processor does not define a separate I/O space. All resources are mapped through the flat 32-bit address space. On-chip I/O devices have their control registers mapped into memory-mapped registers (MMRs) at addresses near the top of the 4G byte address space. These are separated into two smaller blocks, one which contains the control MMRs for all core functions, and the other which contains the registers needed for setup and control of the on-chip peripherals outside of the core. The MMRs are accessible only in supervisor mode and appear as reserved space to on-chip peripherals.

## Memory Protection

The Blackfin cores feature a memory protection concept, which grants data and/or instruction accesses from enabled memory regions only. A supervisor mode vs. user mode programming model supports dynamically varying access rights. Increased flexibility in memory page size options supports a simple method of static memory partitioning.

## Multi-Parity-Bit-Protected L1 Memories

In the processor's L1 memory space, whether SRAM or cache, each word is protected by multiple parity bits to detect the single event upsets that occur in all RAMs. This applies both to L1 instruction and data memory spaces.

## ECC-Protected L2 Memory

Error correcting codes (ECC) are used to correct single event upsets. The L2 memory is protected with a Single Error Correct-Double Error Detect (SEC-DED) code. By default ECC is enabled, but it can be disabled on a per-bank basis. Single-bit errors are transparently corrected. Dual-bit errors can issue a system event or fault if enabled. ECC protection is fully transparent to the user, even if L2 memory is read or written by 8-bit or 16-bit entities.

## CRC-Protected Memories

While parity bit and ECC protection mainly protect against random soft errors in L1 and L2 memory cells, the CRC engines can be used to protect against systematic errors (pointer errors) and static content (instruction code) of L1, L2 and even L3 memories (DDR2, LPDDR). The processors feature two CRC engines which are embedded in the memory-to-memory DMA controllers. CRC check sums can be calculated or compared on the fly during memory transfers, or one or multiple memory regions can be continuously scrubbed by single DMA work unit as per DMA descriptor chain instructions. The CRC engine also protects data loaded during the boot process.

## Watchpoint Protection

The primary purpose of watchpoints and hardware breakpoints is to serve emulator needs. When enabled, they signal an emulator event whenever user-defined system resources are accessed or a core executes from user-defined addresses. Watchdog events can be configured such that they signal the events to the other Blackfin core or to the fault management unit.

## Pin Multiplexing

The processor supports a flexible multiplexing scheme that multiplexes the GPIO pins with various peripherals. A maximum of 4 peripherals plus GPIO functionality is shared by each GPIO pin. All GPIO pins have a bypass path feature - that is, when the output enable and the input enable of a GPIO pin are both active, the data signal before the pad driver is looped back to the receive path for the same GPIO pin.

## Processor Infrastructure

The [ADSP-BF60x Functional Block Diagram](#) shows a number of system control blocks. Some of these blocks provide system control operations, such as event handling and managing the memory sub-system interface. Other of these blocks provide processor core integration (infrastructure), such as bus arbitration, clock generation, and dynamic power management. The following sections provide information on the primary infrastructure components of the ADSP-BF60x processors.

## System Crossbar (SCB)

The system crossbars (SCB) appear as buses in the [ADSP-BF60x Functional Block Diagram](#). These buses constitute a switch fabric, providing concurrent data access between on-chip bus master and slave memory spaces. Each piece of fabric consists of a matrix for each of master interfaces, and each matrix has a number of slave interfaces. Each matrix with programmable round robin arbitration has a matching number of master slots. Arbitration may be programmed by writing each slot register with a corresponding slave value.

## Clock Generation

The clock generation unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a clock that runs at a frequency that is multiple times the CLKIN input clock frequency. It also generates all on-chip clocks and synchronization signals. The PCU allows the application software to control the PLL module operation.

Multiplication factors are programmed to the PLLs to define the PLLCLK frequency. Programmable values divide the PLLCLK frequency to generate the core clock (CCLK), the system clocks (SYSCLK, SCLK0 and SCLK1), the LPDDR or DDR2 clock (DCLK) and the output clock (OCLK). This is illustrated in the [Clock Relationships and Divider Values](#) figure.

Writing to the CGU control registers does not affect the behavior of the PLL immediately. Registers are first programmed with a new value, and the PLL logic executes the changes so that it transitions smoothly from the current conditions to the new ones.

The SYS\_CLKIN pin oscillations start when power is applied to the VDD\_EXT pins. The rising edge of SYS\_HWRST can be applied as soon as all voltage supplies are within specifications, and SYS\_CLKIN oscillations are stable.

A SYS\_CLKOUT output pin has programmable options to output divided-down versions of the on-chip clocks. By default, the SYS\_CLKOUT pin drives a buffered version of the SYS\_CLKIN input. Clock generation faults (for example PLL unlock) may trigger a reset by hardware.

## Crystal Oscillator (SYS\_XTAL)

The processor can be clocked by an external crystal, a sine wave input, or a buffered, shaped clock derived from an external clock oscillator. If an external clock is used, it should be a TTL compatible signal and must not be halted, changed, or operated below the specified frequency during normal operation. This signal is connected to the processor's SYS\_CLKIN pin. When an external clock is used, the SYS\_XTAL pin must be left unconnected. Alternatively, because the processor includes an on-chip oscillator circuit, an external crystal may be used.

## Clock Out/External Clock

The SYS\_CLKOUT pin can be used to output one of several different clocks used on the processor. The clocks shown in [Clock Sources and Dividers](#) can be outputs from SYS\_CLKOUT.

**Table 1-1:** Clock Sources and Dividers

| Clock Source                                                       | Divider                     |
|--------------------------------------------------------------------|-----------------------------|
| CCLK (core clock)                                                  | By 4                        |
| SYSCLK (System clock)                                              | By 2                        |
| SCLK0 (system clock for PVP, all peripherals not covered by SCLK1) | None                        |
| SCLK1 (system clock for SPORTS, SPI, ACM)                          | None                        |
| DCLK (LPDDR/DDR2 clock)                                            | By 2                        |
| OCLK (output clock)                                                | Programmable                |
| CLKBUF                                                             | None, direct from SYS_CLKIN |

## System Protection Unit (SPU)

The system protection unit (SPU) provides features that let you protect system resources from errant writes. A number of protection categories (types of registers to protect) are available.

# Dynamic Power Management (DPM)

The dynamic power management (DPM) feature of the processor lets you control the processor's core clock frequency ( $f_{CCLK}$ ) dynamically.

As shown in clock relationships and divider values figure, the processor supports four different power domains, which maximizes flexibility while maintaining compliance with industry standards and conventions. By isolating the internal logic of the processor into its own power domain, separate from other I/O, the processor can take advantage of dynamic power management without affecting the other I/O devices.

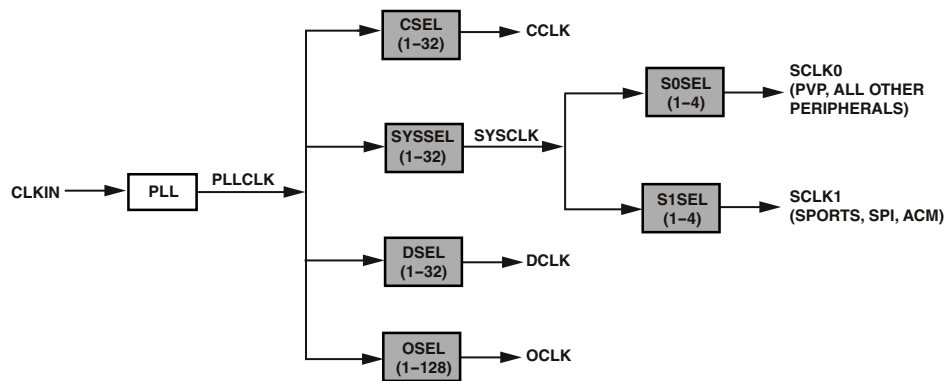


**Figure 1-3:** Clock Relationships and Divider Values

There are no sequencing requirements for the various power domains, but all domains must be powered according to the appropriate [Power Domains and Ranges](#) table for processor operating conditions; even if the feature/peripheral is not used.

**Table 1-2:** Power Domains and Ranges

| Power Domain       | VDD Range     |
|--------------------|---------------|
| All internal logic | $V_{DD\_INT}$ |
| DDR2/LPDDR         | $V_{DD\_DDR}$ |
| USB                | $V_{DD\_USB}$ |
| All other I/O      | $V_{DD\_EXT}$ |

The power dissipated by a processor is largely a function of its clock frequency and the square of the operating voltage. For example, reducing the clock frequency by 25% results in a 25% reduction in dynamic power dissipation.

## Core Timers

Each processor core has its own dedicated timer. The core timer is clocked by the internal processor clock and is typically used as a system tick clock for generating periodic operating system interrupts.

## Event Handling

The processor provides event handling that supports both nesting and prioritization. Nesting allows multiple event service routines to be active simultaneously. Prioritization ensures that servicing of a higher-priority event takes precedence over servicing of a lower-priority event. The processor provides support for five different types of events:

- Emulation - An emulation event causes the processor to enter emulation mode, allowing command and control of the processor via the JTAG interface.
- Reset - This event resets the processor.
- Non maskable Interrupt (NMI) - The NMI event can be generated either by the software watchdog timer, by the NMI input signal to the processor, or by software. The NMI event is frequently used as a power-down indicator to initiate an orderly shutdown of the system.
- Exceptions - Events that occur synchronously to program flow (in other words, the exception is taken before the instruction is allowed to complete). Conditions such as data alignment violations and undefined instructions cause exceptions.
- Interrupts - Events that occur asynchronously to program flow. They are caused by input signals, timers, and other peripherals, as well as by an explicit software instruction.

The [ADSP-BF60x Functional Block Diagram](#) shows a number of system control blocks. Some of these blocks provide system control operations, such as event handling and managing the memory sub-system interface. The following sections provide information on the event handling components of the ADSP-BF60x processors.

### Core Event Controller (CEC)

The CEC supports nine general-purpose interrupts (IVG15-7), in addition to the dedicated interrupt and exception events. Of these general-purpose interrupts, the two lowest-priority interrupts (IVG15-14) are recommended to be reserved for software interrupt handlers. For more information about general-purpose interrupts, see the *Blackfin Processor Programmer's Reference*.

### System Event Controller (SEC)

The SEC manages the enabling, prioritization, and routing of events from each system interrupt or fault source. Additionally, it provides notification and identification of the highest priority active system interrupt request to each core and routes system fault sources to its integrated fault management unit.

## Trigger Routing Unit (TRU)

The TRU provides system-level sequence control without core intervention. The TRU maps trigger masters (generators of triggers) to trigger slaves (receivers of triggers). Slave endpoints can be configured to respond to triggers in various ways. Common applications enabled by the TRU include:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes
- Software triggering
- Synchronization of concurrent activities

## Pin Interrupts

Every port pin on the processor can request interrupts in either an edge-sensitive or a level-sensitive manner with programmable polarity. Interrupt functionality is decoupled from GPIO operation. Six system-level interrupt channels (PINT0-5) are reserved for this purpose. Each of these interrupt channels can manage up to 32 interrupt pins. The assignment from pin to interrupt is not performed on a pin-by-pin basis. Rather, groups of eight pins (half ports) can be flexibly assigned to interrupt channels.

Every pin interrupt channel features a special set of 32-bit memory-mapped registers that enable half-port assignment and interrupt management. This includes masking, identification, and clearing of requests. These registers also enable access to the respective pin states and use of the interrupt latches, regardless of whether the interrupt is masked or not. Most control registers feature multiple MMR address entries to write-one-to-set or write-one-to-clear them individually.

## Memory Architecture

The ADSP-BF60x processor views memory as a single unified 4G byte address space, using 32-bit addresses. All resources, including internal memory, external memory, and I/O control registers, occupy separate sections of this common address space. The memory portions of this address space are arranged in a hierarchical structure to provide a good cost/performance balance of some very fast, low-latency core-accessible memory as cache or SRAM, and larger, lower-cost and performance interface-accessible memory systems.

The processor block diagram shows a number of system control blocks. Some of these blocks provide system control operations, such as event handling and managing the memory sub-system interface. The following sections provide information on managing the memory sub-system interface of the ADSP-BF60x processors.

See the product specific data sheet for the proper external and internal memory configurations.



## Static Memory Controller (SMC)

The static memory controller (SMC) is a protocol converter and data transfer interface between the internal processor bus and the external L3 memory.

The SMC can be programmed to control up to four banks of external memories or memory-mapped devices, with very flexible timing parameters. Each bank occupies a 64M byte segment regardless of the size of the device used, so that these banks are only contiguous if each is fully populated with 64M bytes of memory.

SMC acts as an SCB slave and accesses to SMC are arbitrated by the processor SCB interconnect fabric. On the chip boundary, the SMC is connected to a 25-bit external memory address bus, a 16-bit data bus and memory control signal pins (read, write) including four chip selects via the chip pads. The SMC can support 64 MB of external memory connected to four different banks—each bank is controlled by the chip select signal.

## L2 Memory Controller

The L2 memory controller is an SCB slave with multiple SCB ports that access banks of RAM and/or ROM. These interfaces and the memory subsystem run at the  $L2CLK$  frequency. The L2 memory controller also has a bus interface to provide access to the L2 controller's control and status registers. Each bank of L2 memory is protected by ECC (error correction and control) logic which ensures that single bit errors can be corrected and multi-bit errors can be detected. For each 32KB bank, a 4Kx14 bits ECC RAM stores the parity bits.

## Dynamic Memory Controller (DMC)

The dynamic memory controller (DMC) provides a glue-less interface between DDR2/LPDDR SDRAMs and the system crossbar interface (SCB) on-chip interconnect. This controller supports JESD79-2E compatible double data rate (DDR2) SDRAM and JESD209A low power DDR (LPDDR) SDRAM devices. The DMC enables execution of instructions from (as well as transfer of data to-and-from) external DDR2 SDRAM and external LPDDR SDRAM.

The DMC supports access to the external memory by core and DMA accesses. The external memory address space is divided in to four banks.

The DMC is partitioned in a manner that allows configuration and maintainability. The memory access protocol state machine along with JEDEC standard specific logic is embedded in the protocol controller. An access and operation re-ordering mechanism is incorporated as an efficiency controller. An SCB slave interface is provided to interface with the on-chip interconnect. This interface results in an efficient slave implementation owing to its out-of-order transaction capabilities. The control and status registers present in the DMC controller can be accessed using the MMR access bus.

## Cyclic Redundancy Check (CRC)

The cyclic redundancy check (CRC) peripheral performs the CRC operation of the block of data that is presented to the peripheral. The peripheral provides a means to periodically verify the integrity of the system memory, the contents of memory-mapped registers (MMRs), or communication message objects.

The CRC is a hardware module based on a CRC32 engine that computes the CRC value of the 32-bit data words presented to it. Data is provided by the source channel of the memory-to-memory DMA (in memory scan mode) and is optionally forwarded to the destination channel (memory transfer mode). The main features of the CRC peripheral are:

- Memory scan, memory transfer, data verify, and data fill modes
- User-programmable CRC32 polynomial
- Bit/byte mirroring option (endianness)
- Fault/error interrupt mechanisms
- 1D and 2D fill block to initialize array with constants.
- 32-bit CRC signature of a block of a memory or MMR block.

The dedicated hardware compares the calculated signature of the operation to a pre-loaded expected signature and if the two fail to match, the peripheral generates an error.

Data may be provided by the source channel of the memory-to-memory DMA channels and optionally forwarded to memory via the destination DMA channel. Alternatively, the peripheral also supports data presented by core write transactions.

The CRC peripheral implements a reduced table-lookup algorithm to compute the signature of the data. A programmable 32-bit CRC polynomial is used to automatically generate the lookup table (LUT) contents.

Additional CRC peripheral modes allow for initializing large memory sections with a constant value, or for verifying that sections of memory are equal to a constant value.

The two CRC protection modules allow system software to periodically calculate the signature of code and/or data in memory, the content of memory-mapped registers, or communication message objects. Dedicated hardware circuitry compares the signature with pre calculated values and triggers appropriate fault events.

## Direct Memory Access (DMA)

The direct memory access (DMA) channels are dispersed throughout the infrastructure and may be clustered together over SCBs, sharing a single interface with the main system crossbar.

The processor uses DMA to transfer data within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA controller carries out the data transfers independent of processor activity.

The DMA channels can perform transfers between memory and a peripheral or between one memory and another memory. Two DMA channels are required for memory to memory DMA transfers (MDMA). One channel is the source channel, and the second, the destination channel.

The DMA channel does not connect external memories and devices directly. Rather, data is passed through an external memory interface port. Any kind of device that is supported by the external memory interface can also be accessed by DMA operation. This is typically flash memory, SRAM, DDR SDRAM, FIFOs, or memory-mapped peripheral devices.

All DMAs can transport data to and from all on-chip and off-chip memories. Programs can use two types of DMA transfers, descriptor-based or register-based. Register-based DMA allows the processor to directly program DMA control registers to initiate a DMA transfer. On completion, the control registers may be automatically updated with their original setup values for continuous transfer. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. Descriptor-based DMA transfers allow multiple DMA sequences to be chained together and a DMA channel can be programmed to automatically set up and start another DMA transfer after the current sequence completes.

The DMA controller supports the following DMA operations.

- A single linear buffer that stops on completion.
- A linear buffer with negative, positive or zero stride length.
- A circular, auto-refreshing buffer that interrupts when each buffer becomes full.
- A similar buffer that interrupts on fractional buffers (for example, 1/2, 1/4).
- 1D DMA - uses a set of identical ping-pong buffers defined by a linked ring of two-word descriptor sets, each containing a link pointer and an address.
- 1D DMA - uses a linked list of 4 word descriptor sets containing a link pointer, an address, a length, and a configuration.
- 2D DMA - uses an array of one-word descriptor sets, specifying only the base DMA address.
- 2D DMA - uses a linked list of multi-word descriptor sets, specifying everything.

## On Chip Peripherals

The processor contains a set of on chip peripherals connected to the core over several high-bandwidth buses. These system interface peripherals provide flexibility in system configuration and system performance (see the [ADSP-BF60x Functional Block Diagram](#)). The following sections describe the on chip peripherals that provide the system interface.

## General-Purpose I/O (GPIO)

The general-purpose I/O (GPIO) ports provide the following functions.

- Pin multiplexing scheme
- GPIO functionality
- Pin interrupts

The GPIO port pins can be individually controlled using the port control, status, and interrupt registers. These registers let you:

- Specify the direction of each individual GPIO pin as input or output
- Use the "write one to modify" mechanism to modify any combination of individual GPIO pins in a single instruction, without affecting the level of any other GPIO pins.
- Treat each individual GPIO pin as an interrupt to the processor; GPIO pins defined as inputs can be configured to generate hardware interrupts, while output pins can be triggered by software interrupts
- Specify whether individual pins are level- or edge-sensitive and specify (if edge-sensitive) whether just the rising edge or both the rising and falling edges of the signal are significant

## General-Purpose Timers

The general-purpose timer provides eight general-purpose programmable timers. Each timer has an external pin that can be configured either as a pulse width modulator (PWM) or timer output, as an input to clock the timer, or as a mechanism for measuring pulse widths and periods of external events. These timers can be synchronized to an external clock input on the TMR<sub>x</sub> pins, an external clock TMRCLK input pin, or to the internal SCLK0.

Additionally, a variety of interrupts can be generated upon completion of timer events. Moreover, GP timers can act both as trigger masters and trigger slaves.

The timer units can be used in conjunction with the UARTs and the CAN controller to measure the width of the pulses in the data stream to provide a software auto-baud detect function for the respective serial channels.

## Watchdog Timers

Each core includes a 32-bit timer, which may be used to implement a software watchdog function. A software watchdog can improve system availability by forcing the processor to a known state, via generation of a hardware reset, non maskable interrupt (NMI), or general-purpose interrupt, if the timer expires before being reset by software. The programmer initializes the count value of the timer, enables the appropriate interrupt, then enables the timer. Thereafter, the software must reload the counter before it counts to zero from the programmed value. This protects the system from remaining in an unknown state where

software, which would normally reset the timer, has stopped running due to an external noise condition or software error.

After a reset, software can determine if the watchdog was the source of the hardware reset by interrogating a status bit in the timer control register, which is set only upon a watchdog generated reset.

## General-Purpose Counters

A 32-bit counter is provided that can operate in general-purpose up/down count modes and can sense 2-bit quadrature or binary codes as typically emitted by industrial drives or manual thumbwheels. Count direction is either controlled by a level-sensitive input pin or by two edge detectors.

A third counter input can provide flexible zero marker support and can alternatively be used to input the push-button signal of thumb wheels. All three pins have a programmable debouncing circuit.

Internal signals forwarded to each general-purpose timer enable these timers to measure the intervals between count events. Boundary registers enable auto-zero operation or simple system warning by interrupts when programmable count values are exceeded.

Using this feature, you can convert pulses from incremental position encoders into data that is representative of the actual position by integrating (counting) pulses on one or two inputs. Because integration provides relative position, some devices also feature a zero position input (zero marker) that can be used to establish a reference point to verify that the acquired position does not drift over time. The incremental position information also can be used to determine speed, if the time intervals are measured. The GP counter provides flexible ways to establish position information. When used in conjunction with the GP timer block, the GP counter lets you acquire coherent position and time-stamp information that enables speed calculation.

## Pulsewidth Modulator (PWM)

Each pulsewidth modulator (PWM) block integrates a flexible and programmable 3-phase PWM waveform generator that can be programmed to generate the required switching patterns to drive a 3-phase voltage source inverter for ac induction motor (ACIM) or permanent magnet synchronous motor (PMSM) control. In addition, the PWM block contains special functions that considerably simplify the generation of the required PWM switching patterns for control of the electronically commutated motor (ECM) or brushless dc motor (BDCM). Software can enable a special mode for switched reluctance motors (SRM). The two 3-phase PWM generation units each feature:

- 16-bit center-based PWM generation unit
- Programmable PWM pulse width
- Single/double update modes
- Programmable dead time and switching frequency
- Twos-complement implementation which permits smooth transition to full ON and full OFF states

- Dedicated asynchronous PWM shutdown signal

The eight PWM output signals (per PWM unit) consist of four high-side drive signals and four low-side drive signals. The polarity of a generated PWM signal can be set with software, so that either active HI or active LO PWM patterns can be produced.

Pulses synchronous to the switching frequency can be generated internally and output on the PWM\_SYNC pin. The PWM unit can also accept externally generated synchronization pulses through PWM\_SYNC.

Each PWM unit features a dedicated asynchronous shutdown pin which (when brought low) instantaneously places all six PWM outputs in the OFF state.

## Universal Asynchronous Receiver/Transmitter (UART)

The processors provide two full-duplex universal asynchronous receiver/transmitter (UART) ports, which are fully compatible with PC-standard UARTs. Each UART port provides a simplified UART interface to other peripherals or hosts, supporting full-duplex, DMA-supported, asynchronous transfers of serial data.

A UART port includes support for five to eight data bits, and none, even, or odd parity. Optionally, an additional address bit can be transferred to interrupt only addressed nodes in multi-drop bus (MDB) systems. A frame is terminated by one, one and a half, two or two and a half stop bits. The UARTs also include interrupt-handling hardware. Interrupts can be generated from multiple events.

The UARTs are logically compliant to EIA-232E, EIA-422, EIA-485 and LIN standards, but usually require external transceiver devices to meet electrical requirements. In IrDA (Infrared Data Association) mode, the UARTs meet the half-duplex IrDA SIR (9.6/115.2 Kbps rate) protocol. In multi-drop bus mode, the UARTs meet the full-duplex MDB/ICP v2.0 protocol.

Partial modem status and control functionality is supported by the UART module to allow for hardware flow control. The UART ports support automatic hardware flow control through the Clear To Send (CTS) input and Request To Send (RTS) output with programmable assertion FIFO levels.

To help support the Local Interconnect Network (LIN) protocols, a special command causes the transmitter to queue a break command of programmable bit length into the transmit buffer. Similarly, the number of stop bits can be extended by a programmable inter-frame space.

The capabilities of the UARTs are further extended with support for the Infrared Data Association (IrDA) serial infrared physical layer link specification (SIR) protocol.

The UARTs are DMA-capable peripherals with support for separate transmit and receive DMA master channels. They can be used in either DMA or programmed core mode of operation. The core mode requires software management of the data flow using either interrupts or polling. The DMA method requires minimal software intervention as the DMA engine itself moves the data. Each UART has its own separate transmit and receive DMA channels.

One of the peripheral timers can be used to provide a hardware-assisted auto-baud detection mechanism for use with the UART. The timers are external to the UART.

## 2-Wire Interface (TWI)

The processors include a 2-wire interface (TWI), providing a simple exchange method of control data between multiple devices. The TWI module is compatible with the widely used I<sup>2</sup>C bus standard. The TWI module offers the capabilities of simultaneous master and slave operation and support for both 7-bit addressing and multimedia data arbitration. The TWI interface utilizes two pins for transferring clock (TWI\_SCL) and data (TWI\_SDA) and supports the protocol at speeds up to 400k bits/sec. The TWI interface pins are compatible with 5 V logic levels.

Additionally, the TWI module is fully compatible with serial camera control bus (SCCB) functionality for easier control of various CMOS camera sensor devices.

To preserve processor bandwidth, the TWI module can be set up with transfer initiated interrupts only to service FIFO buffer data reads and writes. Protocol related interrupts are optional. The TWI externally moves 8-bit data while maintaining compliance with the I<sup>2</sup>C bus protocol.

## Controller Area Network (CAN)

The processor includes a controller area network (CAN) module, which implements the CAN 2.0B (active) protocol. This protocol is an asynchronous communications protocol used in both industrial and automotive control systems. The CAN protocol is well suited for control applications due to its capability to communicate reliably over a network. This is because the protocol incorporates CRC checking, message error tracking, and fault node confinement.

**NOTE:** This document assumes reader familiarity with the CAN standard. For more information, refer to Version 2.0 of the CAN Specification from Robert Bosch GmbH.

The CAN module provides the following features:

- 32 mailboxes (8 receive only, 8 transmit only, 16 configurable for receive or transmit)
- Dedicated acceptance masks for each mailbox
- Additional data filtering on first two bytes.
- Support for both the standard (11-bit) and extended (29-bit) identifier (ID) message formats
- Support for remote frames
- Active or passive network support
- CAN wakeup from hibernation mode (lowest static power consumption mode)
- Interrupts, including: TX complete, RX complete, error and global

**NOTE:** An additional crystal is not required to supply the CAN clock, as the CAN clock is derived from a system clock through a programmable divider.

## Universal Serial Bus (USB)

The processor's universal serial bus (USB) module is a USB 2.0 compliant, dual-role device controller. This interface provides a low-cost connectivity solution for the growing adoption of this bus standard in industrial applications and consumer mobile devices, such as cell phones, digital still cameras, and MP3 players. The USB module lets these devices transfer data using a point-to-point USB connection without the need for a PC host.

The USB controller can operate in a traditional USB peripheral-only mode as well as the host mode presented in the on-the-go (OTG) supplement<sup>1</sup> to the USB 2.0 Specification<sup>2</sup>. In host mode, the USB module supports transfers at high-speed (480Mbps), full-speed (12Mbps), and low-speed (1.5Mbps) rates. Peripheral mode supports the high- and full-speed transfer rates.

The USB controller uses a peripheral bus slave interface to access its control and status registers as well as read and write to the endpoint packet buffers. Data is transferred to and from the USB controller through any of the transmit and receive endpoint FIFOs. A DMA bus master interface provides numerous DMA channels to provide a more efficient means of transferring large amounts of data between the controller and the Blackfin processor's memory map.

The USB clock (USB\_CLKIN) is provided through a dedicated external crystal or crystal oscillator. Using an included phase locked loop with programmable multipliers, the USB on-the-go dual-role device controller generates the necessary internal clocking frequency for USB.

## Ethernet Media Access Controller (MAC)

The processor can directly connect to a network by way of an embedded fast Ethernet media access controller (MAC) that supports both 10-BaseT (10M bits/sec) and 100-BaseT (100M bits/sec) operation. The 10/100 Ethernet MAC peripheral on the processor is fully compliant to the IEEE 802.3-2002 standard and it provides programmable features designed to minimize supervision, bus use, or message processing by the rest of the processor system.

Some standard Ethernet MAC features are:

- Support and RMI protocols for external PHYs
- Full duplex and half duplex modes
- Media access management (in half-duplex operation)
- Flow control
- Station management: generation of MDC/MDIO frames for read-write access to PHY registers

---

1. On-The-Go Supplement to the USB 2.0 Specification, Rev 1.0a; June 24, 2003; USB-IF

2. Universal Serial Bus Specification 2.0



Some advanced Ethernet MAC features are:

- Automatic checksum computation of IP header and IP payload fields of Rx frames
- Independent 32-bit descriptor-driven receive and transmit DMA channels
- Frame status delivery to memory through DMA, including frame completion semaphores for efficient buffer queue management in software
- Tx DMA support for separate descriptors for MAC header and payload to eliminate buffer copy operations
- Convenient frame alignment modes
- 47 MAC management statistics counters with selectable clear-on-read behavior and programmable interrupts on half maximum value
- Advanced power management
- Magic packet detection and wakeup frame filtering
- Support for 802.3Q tagged VLAN frames
- Programmable MDC clock rate and preamble suppression

The Ethernet MAC includes support for the IEEE 1588 standard. This standard is a precision clock synchronization protocol for networked measurement and control systems. The processor includes hardware support for IEEE 1588 with an integrated precision time protocol synchronization engine (PTP\_TSYNC). This engine provides hardware assisted time stamping to improve the accuracy of clock synchronization between PTP nodes.

The main IEEE 1588 standard features of the engine are:

- Support for both IEEE 1588-2002 and IEEE 1588-2008 protocol standards
- Hardware assisted time stamping capable of up to 12.5 ns resolution
- Lock adjustment
- Automatic detection of IPv4 and IPv6 packets, as well as PTP messages
- Multiple input clock sources (SCLK0, RMII clock, external clock)
- Programmable pulse per second (PPS) output
- Auxiliary snapshot to time stamp external events

## Removable Storage Interface (RSI)

The removable storage interface (RSI) controller acts as the host interface for multimedia cards (MMC), secure digital memory cards (SD), secure digital input/output cards (SDIO), and CE-ATA hard disk drives. The following list describes the main features of the RSI controller.

- Support for a single MMC, SD memory, SDIO card or CE-ATA hard disk drive
- Support for 1-bit and 4-bit SD modes
- Support for 1-bit, 4-bit, and 8-bit MMC modes
- Support for 4-bit and 8-bit CE-ATA hard disk drives
- Support for eMMC 4.3 embedded NAND flash devices
- A ten-signal external interface with clock, command, and up to eight data lines
- Card interface clock generation from SCLK0
- SDIO interrupt and read wait features
- CE-ATA command completion signal recognition and disable

## Serial Peripheral Interface (SPI)

The processors have two serial peripheral interface (SPI) compatible ports that allow the processor to communicate with multiple SPI compatible devices.

In its simplest mode, the SPI interface uses three pins for transferring data: two data pins (Master Output-Slave Input, MOSI, and Master Input-Slave Output, MISO) and a clock pin (serial clock, SCK). An SPI chip select input pin (SPISS) lets other SPI devices select the processor, and seven SPI chip select output pins (SPISEL7-1) let the processor select other SPI devices. The SPI select pins are reconfigured general-purpose I/O pins. Using these pins, the SPI port provides a full-duplex, synchronous serial interface, which supports both master/slave modes and multi master environments.

The SPI port's baud rate and clock phase/polarities are programmable, and it has integrated DMA channels for both transmit and receive data streams.

## Serial Port (SPORT)

The processor includes three synchronous serial ports (SPORTs) that provide an inexpensive interface to a wide variety of digital and mixed-signal peripheral devices such as the AD183x family of audio CODECs, ADCs, and DACs from Analog Devices. The SPORTs consist of two data lines, a clock, and frame sync. The data lines can be programmed to either transmit or receive and each data line has a dedicated DMA channel.

SPORT data can be automatically transferred to and from on-chip memory/external memory over dedicated DMA channels. Each of the SPORTs can work in conjunction with another SPORT to provide TDM support. In this configuration, one SPORT provides two transmit signals while the other SPORT provides the two receive signals. The frame sync and clock are shared.

Serial ports may operate in any of the following modes:

- Standard DSP serial mode
- Multichannel (TDM) mode
- I<sup>2</sup>S mode
- Packed I<sup>2</sup>S mode
- Left-justified mode

## ADC Control Module (ACM)

The processor includes an ADC control module (ACM) that provides an interface that synchronizes the controls between the processor and an analog-to-digital converter (ADC). The analog-to-digital conversions are initiated by the processor, based on external or internal events.

The ACM permits flexible scheduling of sampling instants and provides precise sampling signals to the ADC.

The ADC, ACM, and SPORT Connections diagram shows how to connect an external ADC to the ACM and one of the three SPORTs.

The ACM synchronizes the ADC conversion process, generating the ADC controls, the ADC conversion start signal, and other signals. The actual data acquisition from the ADC is done by a peripheral, such as a SPORT or a SPI.

The processor interfaces directly to many ADCs without any glue logic required.

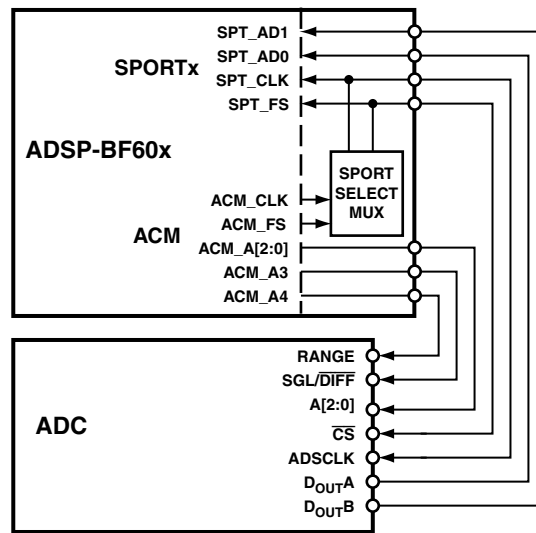


**Figure 1-4:** ADC, ACM, and SPORT Connections

## Link Port (LP)

Four DMA-enabled, 8-bit-wide link ports can connect to the link ports of other Blackfin processors or other link-enabled processors. Link ports are bi-directional ports with eight data lines, an acknowledge line, and a clock line.

## Video Sub-System and Pixel Pipeline (PxP)

The processor's video sub-system provides a connectivity matrix that interconnects the video related peripherals. The pixel pipeline (PxP) protocol is used for data transfer among these video peripherals.

- Three enhanced parallel peripheral interface (PPI) ports
- The pixel compositor (PIXC)
- The pipelined vision processor (PVP)
- The video interconnect (bus) and pixel pipeline (PxP)
- 18 DMA channels

The PPIs can operate in video input and video output modes, and also support several general-purpose modes of operation. These parallel ports are aware of video frame synchronization, blanking concepts and color formats. They can receive up to 16-bit video data directly from video sensors (cameras) and also directly control displays on the output.

The PIXC supports color space conversion and alpha blending for video overlays.

The PVP provides a framework for various vision processing elements, which are targeting mainly edge and object detection strategies.

The video interconnect is a local, distributed bus system, which interconnects the PPI ports, the PIXC, and the PVP.

## Pipelined Vision Processor (PVP)

The pipelined vision processor (PVP) provides hardware implementation of signal and image processing algorithms that are required for co-processing and pre-processing of monochrome video frames in ADAS applications, robotic systems, and other machine applications.

The PVP works in conjunction with the Blackfin cores. It is optimized for convolution and wavelet based object detection and classification, and tracking and verification algorithms. The PVP has the following processing blocks.

- Four 5x5 16-bit convolution blocks optionally followed by downscaling
- A 16-bit cartesian-to-polar coordinate conversion block
- A pixel edge classifier that supports 1st and 2nd derivative modes
- An arithmetic unit with 32-bit addition, multiply and divide
- A 32-bit threshold block with 16 thresholds, a histogram, and run-length encoding
- Two 32-bit integral blocks that support regular and diagonal integrals
- An up- and down-scaling unit with independent scaling ratios for horizontal and vertical components
- Input and output formatters for compatibility with many data formats, including Bayer input format

The PVP can form a pipe of all the constituent algorithmic modules and is dynamically configurable to form different pipeline structures.

The PVP supports the simultaneous processing of up to four data streams. The memory pipe stream operates on data received by DMA from any L1, L2, or L3 memory. The three camera pipe streams operate on a common input received directly from any of the three PPI inputs. Optionally, the PIXC can convert color data received by the PPI and forward luma values to the PVP's monochrome engine. Each stream has a dedicated DMA output. This preprocessing concept ensures careful use of available power and bandwidth budgets and frees up the processor cores for other tasks.

The PVP provides for direct core MMR access to all control/status registers. Two hardware interrupts interface to the system event controller. For optimal performance, the PVP allows register programming through its control DMA interface, as well as outputting selected status registers through the status DMA interface. This mechanism enables the PVP to automatically process job lists completely independent of the Blackfin cores.

## Parallel Peripheral Interface (PPI)

The enhanced parallel peripheral interface (PPI) is a half-duplex, bidirectional port with a dedicated clock pin and three frame sync (FS) pins. It can support direct connections to active TFT LCD, parallel A/D and D/A converters, video encoders and decoders, image sensor modules and other general-purpose peripherals. Each PPI has a DMA channel associated with it. Moreover, in some modes, a PPI may use an additional DMA channel.

The processor provides up to three parallel peripheral interfaces (PPIs), supporting data widths up to 24 bits. The PPI supports direct connection to TFT LCD panels, parallel analog-to-digital and digital-to-analog converters, video encoders and decoders, image sensor modules and other general-purpose peripherals.

The following features are supported in the PPI module:

- Programmable data length: 8 bits, 10 bits, 12 bits, 14 bits, 16 bits, 18 bits, and 24 bits per clock.
- Various framed, non-framed, and general-purpose operating modes. Frame syncs can be generated internally or can be supplied by an external device.
- ITU-656 status word error detection and correction for ITU-656 receive modes and ITU-656 preamble and status word decode.
- Optional packing and unpacking of data to/from 32 bits from/to 8 bits, 16 bits and 24 bits. If packing/unpacking is enabled, endianness can be configured to change the order of packing/unpacking of bytes/words.
- RGB888 can be converted to RGB666 or RGB565 for transmit modes.
- Various de-interleaving/interleaving modes for receiving/transmitting 4:2:2 YCrCb data.
- Configurable LCD data enable (DEN) output available on Frame Sync 3.

## Pixel Compositor (PIXC)

The pixel compositor (PIXC) provides data overlay, transparent color, and color space conversion support. This allows for supporting different video outputs including active (TFT) flat-panel digital color/monochrome LCD displays and analog NTSC/PAL. The color space conversion and text/graphic overlay capabilities, along with visual effect controls, such as transparency control, shorten the processing time on an image data stream, reduce power consumption and save system board space by removing the need for external glue logic.

The PIXC is used to combine and format the data streams required by a wide variety of digital LCD panels and NTSC/PAL analog encoders. It provides all the control needed to allow two data streams from two separate data buffers to be combined and converted into appropriate formats for both LCD panels and video output displays. The main image buffer provides the basic background image presented in the data stream. The overlay image buffer allows the user to add foreground text and graphics on top of the main

image data stream. This feature is useful for printing additional graphical or textual information on the screen, such as symbols or a menu, while showing the main image in the background.

Overlay is an option and can be enabled or disabled. If it is disabled, the blender/compositor is bypassed and the data stream from the main image buffer goes directly to memory with optional color space conversion.

Transparent color is just a special case of blending, masking off the blend operation on a pixel-by-pixel basis. In other words, the overlay region consists of sub-regions in any particular color convenient to the programmer and then, if the color data for a given overlay pixel matches the specified transparent color, the overlay function is masked for that pixel and its data is taken solely from the main image buffer, which is stored in memory in either YUV 4:2:2 interleaved format or RGB888 format

Regardless of the data format or buffer structure, each color element is 8 bits wide. If overlay is enabled, a graphics/text overlay data buffer is defined in memory. The color space converter can switch positions among any of the three locations; it can be in the image data path, the overlay data path, or after the blender. The exact position of the color space converter depends on the input and output data formats.

Since the end display may be a TV (NTSC/PAL) or an LCD panel, and since the image/overlay input buffers may be in either RGB888 or YUV4:2:2 format, a color space conversion may be needed. The color space conversion is selected according to the input data stream format of the PIXC. A YUV-to-RGB format conversion is necessary if the end display is an LCD and if either of the PIXC input data streams is in YUV 4:2:2 format. Similarly, an RGB-to-YUV format conversion is necessary if the end display is a TV and if either of the PIXC input data streams is in RGB888 format.

If the final display device is an LCD, the output RGB data stream is always packed in RGB 8-bit serial format when transferring back to memory. Similarly, if the final display device is a TV, the YUV data stream is always packed in YUV 4:2:2 interleaved format when transferring back to memory.

## Reset Control Unit (RCU)

Reset is the initial state of the whole processor or one of the cores and is the result of a hardware or software triggered event. In this state, all control registers are set to their default values and functional units are idle. Exiting a full system reset starts with Core-0 only being ready to boot. Exiting a Core-n only reset starts with this Core-n being ready to boot.

The reset control unit (RCU) controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. Programs must guarantee that none of the reset functions puts the system into an undefined state or causes resources to stall. This is particularly important when only one of the cores is reset (programs must ensure that there is no pending system activity involving the core that is being reset).

From a system perspective reset is defined by both the reset target and the reset source as described below.

Target defined:

- Hardware Reset - All functional units are set to their default states without exception. History is lost.
- System Reset - All functional units except the RCU are set to their default states.
- Core-n only Reset - Affects Core-n only. The system software should guarantee that the core in reset state is not accessed by any bus master.

Source defined:

- Hardware Reset - The `SYS_HWRST` input signal is asserted active (pulled down).
- System Reset - May be triggered by software (writing to the `RCU_CTL` register) or by another functional unit such as the dynamic power management (DPM) unit (Hibernate) or any of the system event controller (SEC), trigger routing unit (TRU), or emulator. inputs.
- Core-n-only reset - Triggered by software.
- Trigger request (peripheral).

## Booting

The processor has several mechanisms for automatically loading internal and external memory after a reset. The boot mode is defined by the `SYS_BMODE` input pins dedicated for this purpose. There are two categories of boot modes. In master boot modes, the processor actively loads data from parallel or serial memories. In slave boot modes, the processor receives data from external host devices.

The boot modes are shown in the [SYS\\_BMODE Selections and Boot Modes](#) table. These modes are implemented by the `SYS_BMODE` bits of the reset configuration register and are sampled during power-on resets and software-initiated resets.

**Table 1-3:** SYS\_BMODE Selections and Boot Modes

| SYS_BMODE Setting | Boot Mode    |
|-------------------|--------------|
| 000               | No boot/Idle |
| 001               | Flash        |
| 010               | RSI0 Master  |
| 011               | SPI0 Master  |
| 100               | SPI0 Slave   |
| 101               | Reserved     |
| 110               | LP0 Slave    |
| 111               | UART0 Slave  |



## System Design

TBD.

## System Debug Unit

The System Debug Unit (SDU) provides IEEE-1149.1 support through its JTAG interface. In addition to traditional JTAG features, present in legacy Blackfin products, the SDU adds more features for debugging the chip without halting the core processors.

## System Watchpoint Unit

The System Watchpoint Unit (SWU) is a single module which connects to a single system bus and provides for transaction monitoring. One SWU is attached to the bus going to each system slave. The SWU provides ports for all system bus address channel signals. Each SWU contains four match groups of registers with associated hardware. These four SWU match groups operate independently, but share common event (interrupt, trigger, etc.) outputs.



## 2 System Crossbars (SCB)

The System Crossbars (SCB) are the fundamental building blocks of a switch-fabric style system interconnect. The SCBs connect system masters to system slaves and provide concurrent data transfer between multiple masters and multiple slaves. A hierarchical model built from multiple SCBs provides a power and area efficient system interconnect that satisfies the performance and flexibility requirements of a specific system.

### SCB Features

- Highly efficient, pipelined transfer protocol for sustained throughput
- Full-duplex operation for flexibility and reduced latency
- Concurrent transfer support to allow multiple masters to access slaves simultaneously
- Support for protection model (privileged/secure) built into the interconnect
- Programmable arbitration model for bandwidth and latency management

### SCB Functional Description

The following sections provide a functional description of the SCB:

- [SCB Register List](#)
- [SCB Definitions](#)
- [SCB Block Diagram](#)

### SCB Register List

The system cross bar (SCB), which is often referred to as the system interconnect fabric, is a collection of interconnection units connecting system masters to slave memory spaces. Each unit in the fabric consists of a matrix of master interfaces (MIn). Each of these matrices has a number of slave interfaces (SIn). The SCBs (units in the fabric) with multiple SI for each MI have programmable round-robin arbitration to manage access to slots. A set of registers govern SCB operations. For more information on SCB functionality, see the SCB register descriptions.

Table 2-1: SCB Module Registers

| Name        | Description                                           |
|-------------|-------------------------------------------------------|
| SCB_ARBRn   | Arbitration Read Channel Master Interface n Register  |
| SCB_ARBWn   | Arbitration Write Channel Master Interface n Register |
| SCB_SLAVES  | Slave Interfaces Number Register                      |
| SCB_MASTERS | Master Interfaces Number Register                     |

SCB Definitions

To make the best use of the SCB, it is useful to understand the following terms:

*MI (Master Interface)*

SCB master interface connected to system interconnect slave (e.g. L2, sMMR, SCB, etc.).

*SI (Slave Interface)*

SCB slave interface connected to system interconnect master (e.g. Core, DDE, SCB, etc.)

SCB Block Diagram

The SCB architectural model is illustrated in the following figure. A high-level overview of the SCB and associated connections to system masters and slaves is shown. A variable number of masters may be connected to a variable number of slaves in each SCB. In this example all SIs are connected to all MIs as indicated by the lines connecting them:

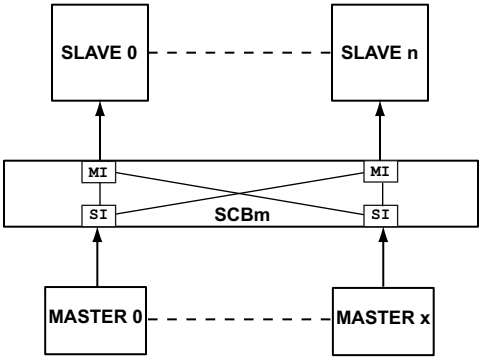


Figure 2-1: SCB Overview

## SCB Hierarchy Block Diagram

A system interconnect built from multiple SCBs in a hierarchical model is illustrated in the following figure. The system master node level SCBs master multiple SIs to a single MI which in turn connects to an SI of the system slave level node SCB. In this example all SIs are connected to all MIs:

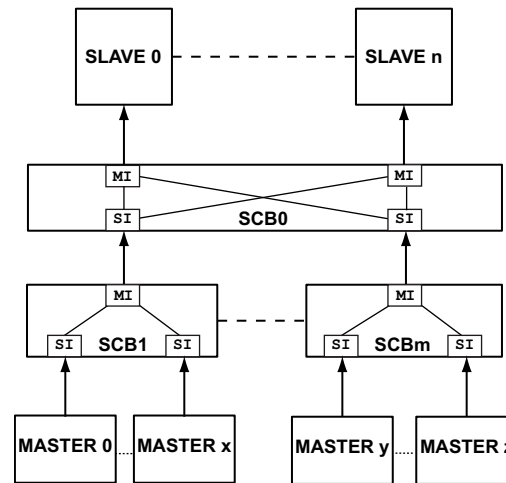


**Figure 2-2:** SCB Hierarchy Overview

## ADSP-BF60x Block Diagram

The following figure shows the block diagram that is specific to the ADSP-BF60x products.

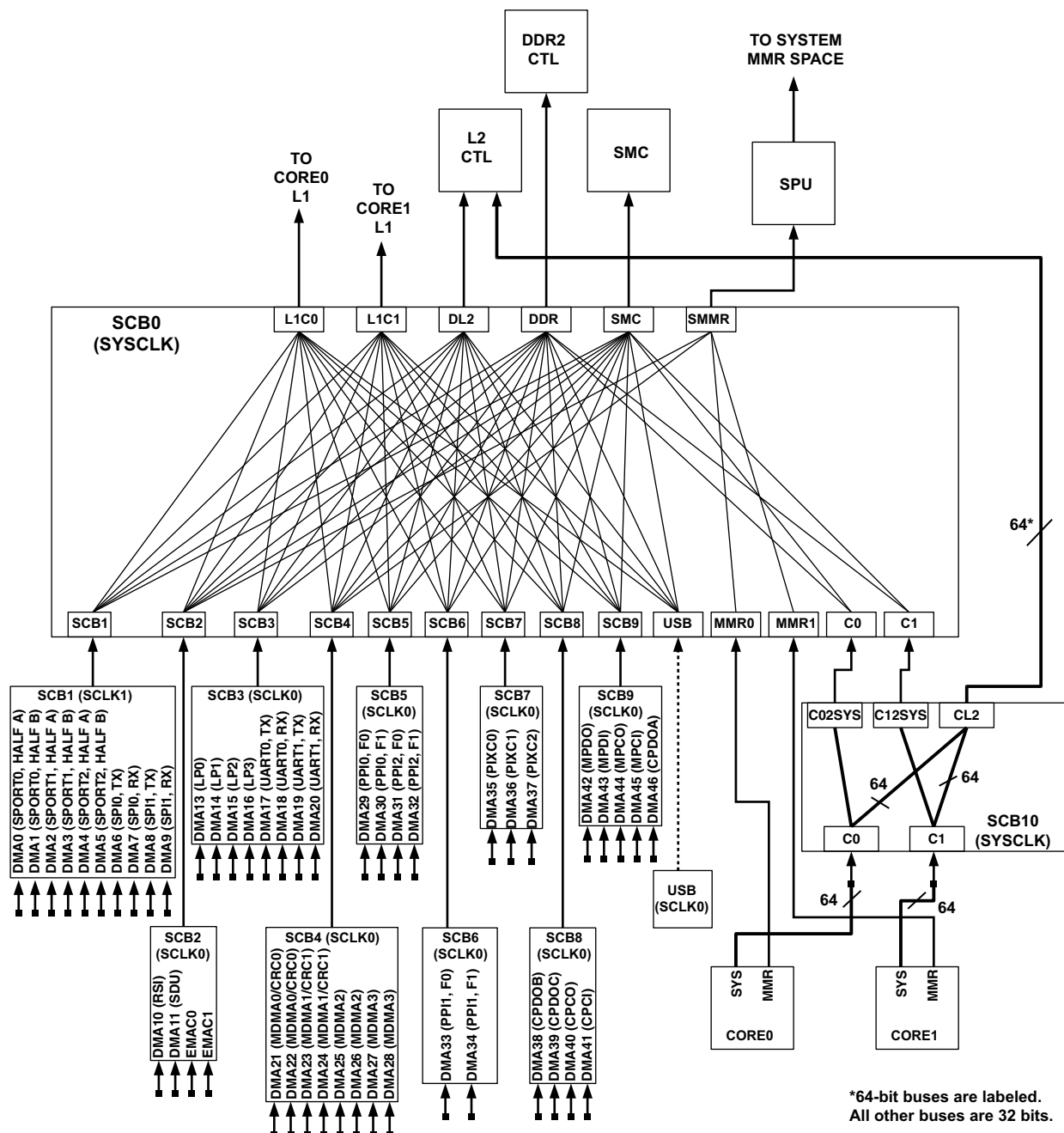


Figure 2-3: ADSP-BF60x Block Diagram.

## SCB Arbitration Tables

The SCB uses round robin arbitration to prioritize each slave's interface to masters (Master Interface) and each master's interface to slaves (Slave Interface). This section provides reference information about these interfaces.

## SCB Units, Master Interfaces, and Arbitration Types

**Table 2-2:** SCB Units, Master Interfaces, and Arbitration Types

| SCB   | Master Interface No. | Arbitration Type  | Programmable Slots |
|-------|----------------------|-------------------|--------------------|
| SCB0  | MI0 (DDR)            | Prog Round Robin  | 32                 |
|       | MI1 (SMC)            | Prog Round Robin  | 32                 |
|       | MI2 (DL2)            | Prog Round Robin  | 32                 |
|       | MI3 (L1C0)           | Prog Round Robin  | 32                 |
|       | MI4 (L1C1)           | Prog Round Robin  | 32                 |
|       | MI5 (SMMR)           | Prog Round Robin  | 16                 |
| SCB1  | MI0                  | Prog Round Robin  | 20                 |
| SCB2  | MI0                  | Prog Round Robin  | 10                 |
| SCB3  | MI0                  | Prog Round Robin  | 16                 |
| SCB4  | MI0                  | Prog Round Robin  | 16                 |
| SCB5  | MI0                  | Prog Round Robin  | 8                  |
| SCB6  | MI0                  | Prog Round Robin  | 4                  |
| SCB7  | MI0                  | Prog Round Robin  | 6                  |
| SCB8  | MI0                  | Prog Round Robin  | 8                  |
| SCB9  | MI0                  | Prog Round Robin  | 10                 |
| SCB10 | MI0 (CL2)            | Prog Round Robin  | 16                 |
|       | MI1 (C02Sys)         | Fixed Round Robin | 0                  |
|       | MI2 (C12Sys)         | Fixed Round Robin | 0                  |

## SCB0 Slots and Masters

**Table 2-3:** SCB0 Arbitration Default Settings for MI0 (DDR)

| Slot | Master      |
|------|-------------|
| 0    | SI0 (C0)    |
| 1    | SI2 (C1)    |
| 2    | SI4 (SCB9)  |
| 3    | SI5 (SCB5)  |
| 4    | SI6 (SCB6)  |
| 5    | SI7 (SCB8)  |
| 6    | SI8 (SCB7)  |
| 7    | SI9 (SCB1)  |
| 8    | SI10 (SCB2) |
| 9    | SI11 (SCB3) |
| 10   | SI13 (USB)  |
| 11   | SI12 (SCB4) |
| 12   | SI0 (C0)    |
| 13   | SI2 (C1)    |
| 14   | SI4 (SCB9)  |
| 15   | SI5 (SCB5)  |
| 16   | SI6 (SCB6)  |
| 17   | SI7 (SCB8)  |
| 18   | SI8 (SCB7)  |
| 19   | SI9 (SCB1)  |
| 20   | SI10 (SCB2) |



**Table 2-3:** SCB0 Arbitration Default Settings for MI0 (DDR) (Continued)

| Slot | Master      |
|------|-------------|
| 21   | SI11 (SCB3) |
| 22   | SI13 (USB)  |
| 23   | SI12 (SCB4) |
| 24   | SI0 (C0)    |
| 25   | SI2 (C1)    |
| 26   | SI4 (SCB9)  |
| 27   | SI5 (SCB5)  |
| 28   | SI6 (SCB6)  |
| 29   | SI7 (SCB8)  |
| 30   | SI8 (SCB7)  |
| 31   | SI13 (USB)  |

**Table 2-4:** SCB0 Arbitration Default Settings for MI1 (SMC)

| Slot | Master      |
|------|-------------|
| 0    | SI0 (C0)    |
| 1    | SI2 (C1)    |
| 2    | SI4 (SCB9)  |
| 3    | SI5 (SCB5)  |
| 4    | SI6 (SCB6)  |
| 5    | SI7 (SCB8)  |
| 6    | SI8 (SCB7)  |
| 7    | SI9 (SCB1)  |
| 8    | SI10 (SCB2) |

**Table 2-4:** SCB0 Arbitration Default Settings for MI1 (SMC) (Continued)

| Slot | Master      |
|------|-------------|
| 9    | SI11 (SCB3) |
| 10   | SI13 (USB)  |
| 11   | SI12 (SCB4) |
| 12   | SI0 (C0)    |
| 13   | SI2 (C1)    |
| 14   | SI4 (SCB9)  |
| 15   | SI5 (SCB5)  |
| 16   | SI6 (SCB6)  |
| 17   | SI7 (SCB8)  |
| 18   | SI8 (SCB7)  |
| 19   | SI9 (SCB1)  |
| 20   | SI10 (SCB2) |
| 21   | SI11 (SCB3) |
| 22   | SI13 (USB)  |
| 23   | SI12 (SCB4) |
| 24   | SI0 (C0)    |
| 25   | SI2 (C1)    |
| 26   | SI4 (SCB9)  |
| 27   | SI5 (SCB5)  |
| 28   | SI6 (SCB6)  |
| 29   | SI7 (SCB8)  |
| 30   | SI8 (SCB7)  |

**Table 2-4:** SCB0 Arbitration Default Settings for MI1 (SMC) (Continued)

| Slot | Master     |
|------|------------|
| 31   | SI13 (USB) |

**Table 2-5:** SCB0 Arbitration Default Settings for MI2 (DL2)

| Slot | Master      |
|------|-------------|
| 0    | SI4 (SCB9)  |
| 1    | SI5 (SCB5)  |
| 2    | SI6 (SCB6)  |
| 3    | SI7 (SCB8)  |
| 4    | SI8 (SCB7)  |
| 5    | SI9 (SCB1)  |
| 6    | SI10 (SCB2) |
| 7    | SI11 (SCB3) |
| 8    | SI13 (USB)  |
| 9    | SI12 (SCB4) |
| 10   | SI4 (SCB9)  |
| 11   | SI5 (SCB5)  |
| 12   | SI6 (SCB6)  |
| 13   | SI7 (SCB8)  |
| 14   | SI8 (SCB7)  |
| 15   | SI9 (SCB1)  |
| 16   | SI10 (SCB2) |
| 17   | SI11 (SCB3) |
| 18   | SI13 (USB)  |

**Table 2-5:** SCB0 Arbitration Default Settings for MI2 (DL2) (Continued)

| Slot | Master      |
|------|-------------|
| 19   | SI12 (SCB4) |
| 20   | SI4 (SCB9)  |
| 21   | SI5 (SCB5)  |
| 22   | SI6 (SCB6)  |
| 23   | SI7 (SCB8)  |
| 24   | SI8 (SCB7)  |
| 25   | SI9 (SCB1)  |
| 26   | SI10 (SCB2) |
| 27   | SI11 (SCB3) |
| 28   | SI13 (USB)  |
| 29   | SI12 (SCB4) |
| 30   | SI4 (SCB9)  |
| 31   | SI7 (SCB8)  |

**Table 2-6:** MI3 (L1A)

| Slot | Master      |
|------|-------------|
| 0    | SI4 (SCB9)  |
| 1    | SI5 (SCB5)  |
| 2    | SI6 (SCB6)  |
| 3    | SI7 (SCB8)  |
| 4    | SI8 (SCB7)  |
| 5    | SI9 (SCB1)  |
| 6    | SI10 (SCB2) |

**Table 2-6:** MI3 (L1A) (Continued)

| Slot | Master      |
|------|-------------|
| 7    | SI11 (SCB3) |
| 8    | SI13 (USB)  |
| 9    | SI12 (SCB4) |
| 10   | SI4 (SCB9)  |
| 11   | SI5 (SCB5)  |
| 12   | SI6 (SCB6)  |
| 13   | SI7 (SCB8)  |
| 14   | SI8 (SCB7)  |
| 15   | SI9 (SCB1)  |
| 16   | SI10 (SCB2) |
| 17   | SI11 (SCB3) |
| 18   | SI13 (USB)  |
| 19   | SI12 (SCB4) |
| 20   | SI4 (SCB9)  |
| 21   | SI5 (SCB5)  |
| 22   | SI6 (SCB6)  |
| 23   | SI7 (SCB8)  |
| 24   | SI8 (SCB7)  |
| 25   | SI9 (SCB1)  |
| 26   | SI10 (SCB2) |
| 27   | SI11 (SCB3) |
| 28   | SI13 (USB)  |

**Table 2-6:** MI3 (L1A) (Continued)

| Slot | Master      |
|------|-------------|
| 29   | SI12 (SCB4) |
| 30   | SI4 (SCB9)  |
| 31   | SI7 (SCB8)  |

**Table 2-7:** MI4 (L1B)

| Slot | Master      |
|------|-------------|
| 0    | SI4 (SCB9)  |
| 1    | SI5 (SCB5)  |
| 2    | SI6 (SCB6)  |
| 3    | SI7 (SCB8)  |
| 4    | SI8 (SCB7)  |
| 5    | SI9 (SCB1)  |
| 6    | SI10 (SCB2) |
| 7    | SI11 (SCB3) |
| 8    | SI13 (USB)  |
| 9    | SI12 (SCB4) |
| 10   | SI4 (SCB9)  |
| 11   | SI5 (SCB5)  |
| 12   | SI6 (SCB6)  |
| 13   | SI7 (SCB8)  |
| 14   | SI8 (SCB7)  |
| 15   | SI9 (SCB1)  |
| 16   | SI10 (SCB2) |

**Table 2-7:** MI4 (L1B) (Continued)

| Slot | Master      |
|------|-------------|
| 17   | SI11 (SCB3) |
| 18   | SI13 (USB)  |
| 19   | SI12 (SCB4) |
| 20   | SI4 (SCB9)  |
| 21   | SI5 (SCB5)  |
| 22   | SI6 (SCB6)  |
| 23   | SI7 (SCB8)  |
| 24   | SI8 (SCB7)  |
| 25   | SI9 (SCB1)  |
| 26   | SI10 (SCB2) |
| 27   | SI11 (SCB3) |
| 28   | SI13 (USB)  |
| 29   | SI12 (SCB4) |
| 30   | SI4 (SCB9)  |
| 31   | SI7 (SCB8)  |

**Table 2-8:** MI5 (SMMR)

| Slot | Master      |
|------|-------------|
| 0    | SI1 (MMR0)  |
| 1    | SI3 (MMR1)  |
| 2    | SI10 (SCB2) |
| 3    | SI12 (SCB4) |
| 4    | SI1 (MMR0)  |

**Table 2-8:** MI5 (SMMR) (Continued)

| Slot | Master      |
|------|-------------|
| 5    | SI3 (MMR1)  |
| 6    | SI10 (SCB2) |
| 7    | SI12 (SCB4) |
| 8    | SI1 (MMR0)  |
| 9    | SI3 (MMR1)  |
| 10   | SI10 (SCB2) |
| 11   | SI12 (SCB4) |
| 12   | SI1 (MMR0)  |
| 13   | SI3 (MMR1)  |
| 14   | SI10 (SCB2) |
| 15   | SI12 (SCB4) |

## SCB1 Slots and Masters

**Table 2-9:** MI0 (SCB1)

| Slot | Master        |
|------|---------------|
| 0    | SI0 (SPORT0A) |
| 1    | SI1 (SPORT0B) |
| 2    | SI2 (SPORT1A) |
| 3    | SI3 (SPORT1B) |
| 4    | SI4 (SPORT2A) |
| 5    | SI5 (SPORT2B) |
| 6    | SI6 (SPI0TX)  |
| 7    | SI7 (SPI0RX)  |



**Table 2-9:** MIO (SCB1) (Continued)

| Slot | Master        |
|------|---------------|
| 8    | SI8 (SPI1TX)  |
| 9    | SI9 (SPI1RX)  |
| 10   | SI0 (SPORT0A) |
| 11   | SI1 (SPORT0B) |
| 12   | SI2 (SPORT1A) |
| 13   | SI3 (SPORT1B) |
| 14   | SI4 (SPORT2A) |
| 15   | SI5 (SPORT2B) |
| 16   | SI6 (SPI0TX)  |
| 17   | SI7 (SPI0RX)  |
| 18   | SI8 (SPI1TX)  |
| 19   | SI9 (SPI1RX)  |

## SCB2 Slots and Masters

**Table 2-10:** MIO (SCB2)

| Slot | Master        |
|------|---------------|
| 0    | SI0 (RSI)     |
| 1    | SI1 (SDU DMA) |
| 2    | SI2 (SDU)     |
| 3    | SI3 (EMAC0)   |
| 4    | SI4 (EMAC1)   |
| 5    | SI0 (RSI)     |
| 6    | SI1 (SDU DMA) |

**Table 2-10:** MI0 (SCB2) (Continued)

| Slot | Master      |
|------|-------------|
| 7    | SI2 (SDU)   |
| 8    | SI3 (EMAC0) |
| 9    | SI4 (EMAC1) |

## SCB3 Slots and Masters

**Table 2-11:** MI0 (SCB3)

| Slot | Master        |
|------|---------------|
| 0    | SI0 (LP0)     |
| 1    | SI1 (LP1)     |
| 2    | SI2 (LP2)     |
| 3    | SI3 (LP3)     |
| 4    | SI4 (UART0TX) |
| 5    | SI5 (UART0RX) |
| 6    | SI6 (UART1TX) |
| 7    | SI7 (UART1RX) |
| 8    | SI0 (LP0)     |
| 9    | SI1 (LP1)     |
| 10   | SI2 (LP2)     |
| 11   | SI3 (LP3)     |
| 12   | SI4 (UART0TX) |
| 13   | SI5 (UART0RX) |
| 14   | SI6 (UART1TX) |
| 15   | SI7 (UART1RX) |

## SCB4 Slots and Masters

**Table 2-12:** MI0 (SCB4)

| Slot | Master      |
|------|-------------|
| 0    | SI0 (DMA21) |
| 1    | SI1 (DMA22) |
| 2    | SI2 (DMA23) |
| 3    | SI3 (DMA24) |
| 4    | SI4 (DMA25) |
| 5    | SI5 (DMA26) |
| 6    | SI6 (DMA27) |
| 7    | SI7 (DMA28) |
| 8    | SI0 (DMA21) |
| 9    | SI1 (DMA22) |
| 10   | SI2 (DMA23) |
| 11   | SI3 (DMA24) |
| 12   | SI4 (DMA25) |
| 13   | SI5 (DMA26) |
| 14   | SI6 (DMA27) |
| 15   | SI7 (DMA28) |

## SCB5 Slots and Masters

**Table 2-13:** MI0 (SCB5)

| Slot | Master              |
|------|---------------------|
| 0    | SI0 (PPI0 -- DMA29) |
| 1    | SI1 (PPI0 -- DMA30) |

**Table 2-13:** MI0 (SCB5) (Continued)

| Slot | Master              |
|------|---------------------|
| 2    | SI2 (PPI2 -- DMA31) |
| 3    | SI3 (PPI2 -- DMA32) |
| 4    | SI0 (PPI0 -- DMA29) |
| 5    | SI1 (PPI0 -- DMA30) |
| 6    | SI2 (PPI2 -- DMA31) |
| 7    | SI3 (PPI2 -- DMA32) |

## SCB6 Slots and Masters

**Table 2-14:** MI0 (SCB6)

| Slot | Master              |
|------|---------------------|
| 0    | SI0 (PPI1 -- DMA33) |
| 1    | SI1 (PPI1 -- DMA34) |
| 2    | SI0 (PPI1 -- DMA33) |
| 3    | SI1 (PPI1 -- DMA34) |

## SCB7 Slots and Masters

**Table 2-15:** MI0 (SCB7)

| Slot | Master      |
|------|-------------|
| 0    | SI0 (PIXC0) |
| 1    | SI1 (PIXC1) |
| 2    | SI2 (PIXC2) |
| 3    | SI0 (PIXC0) |
| 4    | SI1 (PIXC1) |
| 5    | SI2 (PIXC2) |

## SCB8 Slots and Masters

**Table 2-16:** MI0 (SCB8)

| Slot | Master           |
|------|------------------|
| 0    | SI0 ( PVP CPDOB) |
| 1    | SI1 (PVP CPDOC)  |
| 2    | SI2 (PVP CPCO)   |
| 3    | SI3 (PVP CPCI)   |
| 4    | SI0 (PVP CPDOB)  |
| 5    | SI1 (PVP CPDOC)  |
| 6    | SI2 (PVP CPCO)   |
| 7    | SI3 (PVP CPCI)   |

## SCB9 Slots and Masters

**Table 2-17:** MI0 (SCB9)

| Slot | Master              |
|------|---------------------|
| 0    | SI0 (PVP MPDO/UDDO) |
| 1    | SI1 (PVP MPDI/UDDI) |
| 2    | SI2 (PVP MPCO/UPCO) |
| 3    | SI3 (PVP MPCI/UPCI) |
| 4    | SI4 (PVP CPDOA)     |
| 5    | SI0 (PVP MPDO/UDDO) |
| 6    | SI1 (PVP MPDI/UDDI) |
| 7    | SI2 (PVP MPCO/UPCO) |
| 8    | SI3 (PVP MPCI/UPCI) |
| 9    | SI4 (PVP CPDOA)     |

## SCB Programming Model

### SCB Register Access Restrictions

The following information applies to the SCB registers:

1. Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unpredictable behavior of the SCB
2. Unless otherwise stated in the accompanying text:
  - Do not modify undefined register bits
  - Ignore undefined register bits on reads
  - All register bits are reset to a logic 0 by a system reset

### SCB Programming Concepts

#### Arbitration Model Configuration

The arbitration model of the SCB is programmable. Through register configuration, the arbitration model can be adjusted for the specific bandwidth and latency requirements. The arbitration model is programmable round robin with the following features:

- Each MI has its own set of arbitration slots
- Total number of arbitration slots is specific to the SCB and fixed in hardware
- Arbitration slot priority rotates each cycle
- SIs are assigned to each slot (see programming model)
- Transactions are granted to highest priority requesting master each cycle

## SCB Programming Examples

Each MI of an SCB has an Arbitration Write Channel register and an Arbitration Read Channel register which provide indexed access to the write channel and read channel arbitration parameters respectively for the MI. of the slave interfaces (SIn) connected to the SCB.

### Reading Arbitration Settings

To read arbitration parameter data (for example, get the identity of the SI assigned to an arbitration slot), use a write-followed-by-read process with bits 31-24 set to FF:

1. Write SCB\_ARBWn with the value 0xFFmm (where mm is the slot index for the following read)
2. Read the value in SCB\_ARBWn, the value is 0x00nn (where nn is identity of the SI assigned to slot mm)

### Writing Arbitration Settings

To write arbitration parameter data (for example, assign an SI to an SCB slot), use a write with bits 31-24 set to the slot number. For information about the SCBs in the processor and their slot numbers, see the SCB functional description section.

**NOTE:** No protection is imposed when you program the slots in the arbitration scheme. It is possible to remove an SI from all the slots (making the SI inaccessible).

## SCB Register Descriptions

System Cross Bar (SCB) contains the following registers.

**Table 2-18:** SCB Module Registers

| Name        | Description                                           |
|-------------|-------------------------------------------------------|
| SCB_ARBRn   | Arbitration Read Channel Master Interface n Register  |
| SCB_ARBWn   | Arbitration Write Channel Master Interface n Register |
| SCB_SLAVES  | Slave Interfaces Number Register                      |
| SCB_MASTERS | Master Interfaces Number Register                     |

### Arbitration Read Channel Master Interface n Register

Each master interface (MI) of an SCB has an SCB\_ARBRn register, providing indexed access to the read arbitration parameters of the slave interfaces (SIn) connected to that MI.

SCB\_ARBRn: Arbitration Read Channel Master Interface n Register - R/W

Reset = 0x0000 0000

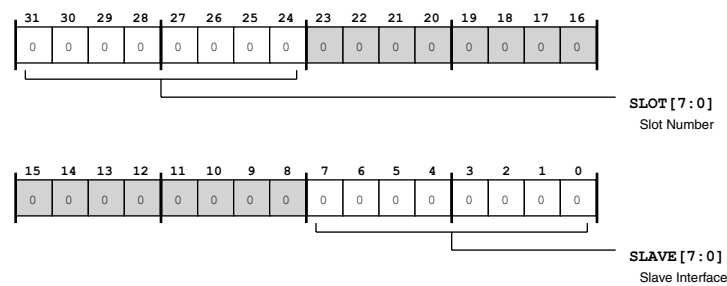


**Figure 2-4:** SCB\_ARBRn Register Diagram

**Table 2-19:** SCB\_ARBRn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:24   | SLOT     | Slot Number.<br>The SCB_ARBRn . SLOT bits either hold the SCB slot number (for writing arbitration data) or hold the value FF (for write-followed-by-read of arbitration data). For a list of slot numbers for specific SCBs, see the SCB function description.                                                                         |
| 7:0     | SLAVE    | Slave Interface.<br>The SCB_ARBRn . SLAVE bits either hold the SCB slave interface (SI) to be assigned to the slot number in SCB_ARBRn . SLOT (for writing arbitration data) or hold the slot number (for write-followed-by-read of arbitration data). For a list of slot numbers of specific SCBs, see the SCB functional description. |

Arbitration Write Channel Master Interface n Register

Each master interface (MI) SCB has an SCB\_ARBWn register, providing indexed access to the write arbitration parameters of the slave interfaces (SIn) connected to that MI.



**SCB\_ARBWn: Arbitration Write Channel Master Interface n Register - R/W**

Reset = 0x0000 0000

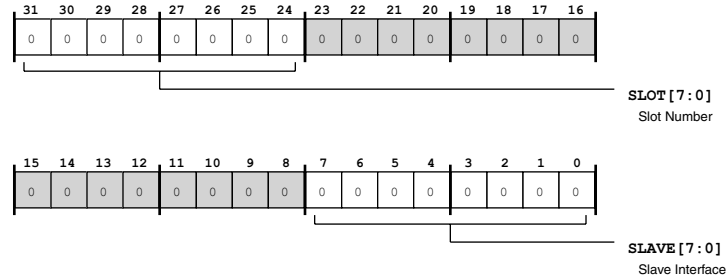


**Figure 2-5:** SCB\_ARBWn Register Diagram

**Table 2-20:** SCB\_ARBWn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:24   | SLOT     | Slot Number.<br>The SCB_ARBWn.SLOT bits either hold the SCB slot number (for writing arbitration data) or hold the value FF (for write-followed-by-read of arbitration data). For a list of slot numbers for specific SCBs, see the SCB function description.                                                                       |
| 7:0     | SLAVE    | Slave Interface.<br>The SCB_ARBWn.SLAVE bits either hold the SCB slave interface (SI) to be assigned to the slot number in SCB_ARBWn.SLOT (for writing arbitration data) or hold the slot number (for write-followed-by-read of arbitration data). For a list of slot numbers of specific SCBs, see the SCB functional description. |

## Slave Interfaces Number Register

The SCB\_SLAVES register holds the number of slave interfaces (SI) connected to the SCB.

**SCB\_SLAVES: Slave Interfaces Number Register - R/NW**

Reset = 0x0000 0000

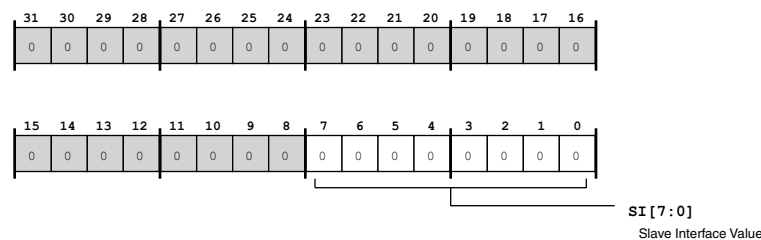


**Figure 2-6:** SCB\_SLAVES Register Diagram

**Table 2-21:** SCB\_SLAVES Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                    |
|---------|----------|------------------------------------------------------------------------------------------------------------|
| 7:0     | SI       | Slave Interface Value.<br>The SCB_SLAVES.SI bits hold the number of slave interfaces connected to the SCB. |

Master Interfaces Number Register

The SCB\_MASTERS register holds the number of master interfaces (SI) connected to the SCB.

**SCB\_MASTERS: Master Interfaces Number Register - R/NW**

Reset = 0x0000 0000

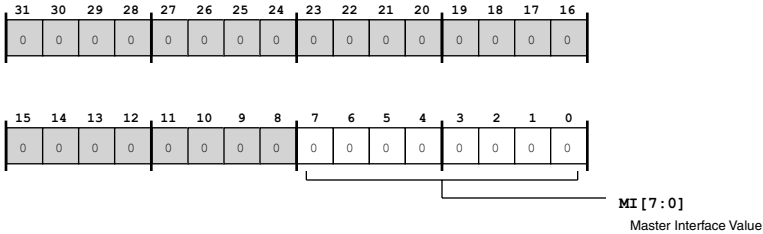


**Figure 2-7:** SCB\_MASTERS Register Diagram

**Table 2-22:** SCB\_MASTERS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------|
| 7:0     | MI       | Master Interface Value.<br>The SCB_MASTERS.MI bits hold the number of master interfaces connected to the SCB. |





## 3 Clock Generation Unit (CGU)

The Clock Generation Unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a clock that runs at a frequency that is multiple times the CLKIN input clock frequency. It also generates all on-chip clocks and synchronization signals. The PCU allows the application software to control the PLL module operation.

### CGU Functional Description

The CGU (clock generation unit) generates all on-chip clocks and synchronization signals based on the programmed PLL multiplication factor and dividers.

### CGU Register List

The clock generation unit (CGU) includes the phase locked loop (PLL) and the PLL control unit (PCU). The PLL generates a clock, running at a frequency that is a multiple of the CLKIN input clock's frequency. The CGU also generates all on-chip clocks and synchronization signals. The PCU permits application software control of the PLL's operation. A set of registers govern CGU operations. For more information on CGU functionality, see the CGU register descriptions.

**Table 3-1:** CGU Module Registers

| Name          | Description            |
|---------------|------------------------|
| CGU_CTL       | Control Register       |
| CGU_STAT      | Status Register        |
| CGU_DIV       | Divisor Register       |
| CGU_CLKOUTSEL | CLKOUT Select Register |

## ADSP-BF60x CGU Interrupt List

**Table 3-2:** ADSP-BF60x CGU Interrupt List

| Description | ID  | DMA Channel | Sensitivity |
|-------------|-----|-------------|-------------|
| CGU0 Event  | 1   |             | PULSE/EDGE  |
| CGU0 Error  | 129 |             | LEVEL       |

## ADSP-BF60x CGU Trigger List

**Table 3-3:** Masters ADSP-BF60x CGU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
| CGU0 Event  | 1  | PULSE/EDGE  |
|             |    |             |

**Table 3-4:** Slaves ADSP-BF60x CGU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

## CGU Definitions

### CGU

Clock Generation Unit which is comprised of the PLL and PCU. The CGU generates the following clocks:

**Table 3-5:** Clock Descriptions

| Clock  | Description                                                                                 |
|--------|---------------------------------------------------------------------------------------------|
| PLLCLK | Provides the source from which all clocks below are derived from unless the PLL is bypassed |
| CCLK0  | Core Clock 0                                                                                |
| CCLK1  | Core Clock 1                                                                                |
| SYSCLK | Clock for system buses and provides the source from which SCLK0 and SCLK1 are derived       |
| SCLK0  | PVP and all other peripherals not clocked by SCLK1                                          |
| SCLK1  | SPORT, SPI, and ACM peripherals clock                                                       |
| DCLK   | Dynamic memory clock                                                                        |
| OUTCLK | OUTCLK is a possible source for SYS_CLKOUT                                                  |

### DPM

Dynamic Power Management Controller

### PCU

PLL control unit

### PLL

Phase locked loop

### RCU

Reset control unit

## CGU PLL Block Diagram

The following figure is a top level block diagram of the phase locked loop (PLL). The main blocks of the PLL are the phase frequency detector (PFD), the charge pump, the loop filter, and the voltage controlled

oscillator (VCO) which multiplies SYS\_CLKIN to a higher frequency. The output of these blocks is called PLLCLK. The PLLCLK is divided to form CCLK0, CCLK1, SYSCCLK, DCLK, and OUTCLK. The SYSCCLK is further divided to form SCLK0 and SCLK1. The OUTCLK is routed to the CLKOUT module shown below so it can be selected as one of the SYS\_CLKOUT sources.

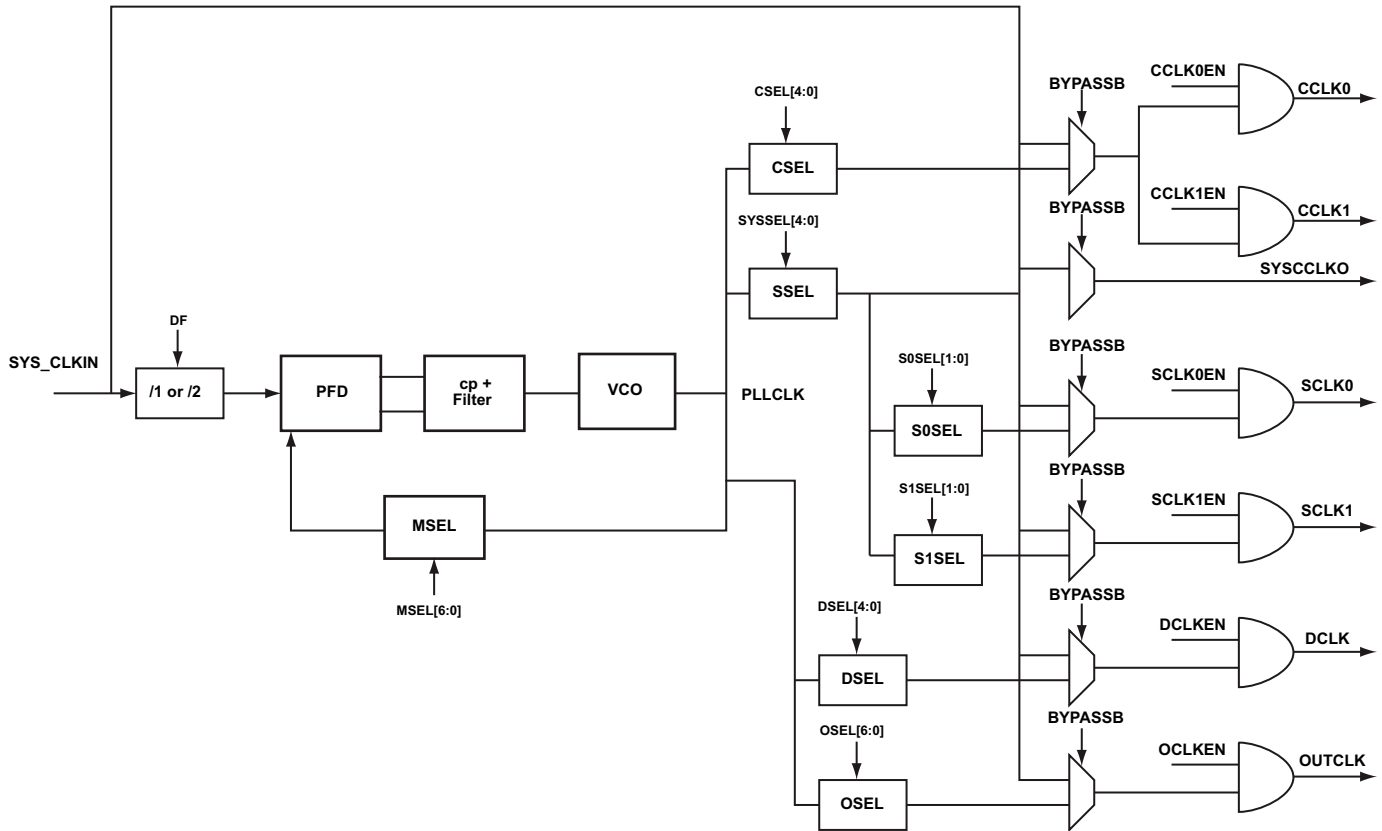


**Figure 3-1: PLL Top Level Block Diagram**

The following figure is a conceptual representation of the CLKOUT module. As shown in the PLL Top Level Block Diagram, many clocks are available on the output pin **SYS\_CLKOUT**. The selection of which clock outputs on **SYS\_CLKOUT** is controlled by **CGU\_CLKOUTSEL.CLKOUTSEL**.



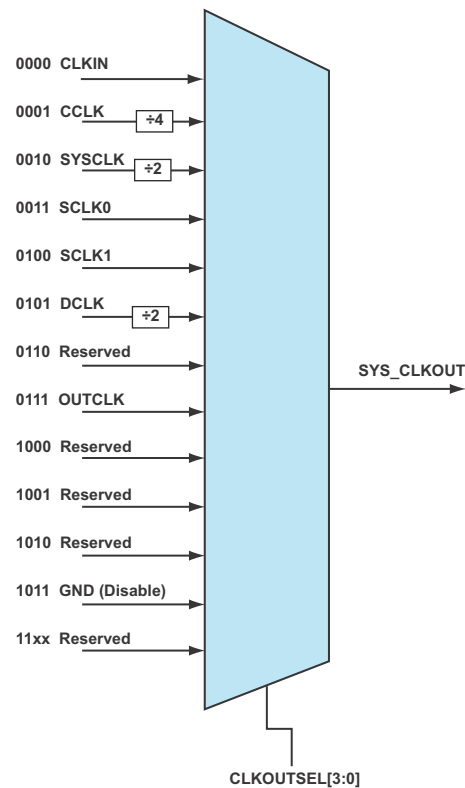


**Figure 3-2:** CLKOUT Module

## CGU Event Control

The CGU is capable of generating a CGU Event or CGU Error for several different causes.

### CGU Event

The CGU Event interrupt is used for different purposes. After a frequency change, a CGU Event indicates that the PLL has locked and clocks are synchronized. The CGU Event interrupt can be used to break the core idle if the core was idled while change frequencies. While in ACTIVE mode, a CGU Event indicates that the PLL has locked. Before boot the CGU Event is used to signal the RCU to kick off the boot process.

### CGU Error

If the PLL fails to lock, the PLL will be disabled and a CGU Error is triggered. In addition, `CGU_STAT.PLOCKERR` is set. The PLL resets itself when it is disabled. If the lock error happens during reset, system reset will be exited and a CGU Event will be triggered. If the failed lock occurs during a frequency change the cores exit idle. In order to clear the CGU Event in an Interrupt Service Routine, write to

`CGU_CTL.MSEL` or `CGU_CTL.DF`. This makes the PLL exit the error state (`CGU_STAT.PLOCKERR` is cleared and the `CGU_EVENT` is de-asserted) and re-lock. This occurs even if the new `CGU_CTL.MSEL` and `CGU_CTL.DF` values are the same as the old values. `CGU_CTL.LOCK` and `CGU_CTL.WFI` settings will still apply. If the PLL lock error occurred during a hardware or software triggered system reset, once out of reset the system is functional and can boot. If the PLL lock error occurs during a PLL frequency change, the CGU Event interrupt makes the cores exit idle, as in the non error case.

The `CGU_STAT.WDIVERR` indicates a write access to the `CGU_DIV` register (to trigger an alignment sequence or to change `CGU_DIV.CSEL`, `CGU_DIV.SYSSEL`, `CGU_DIV.S0SEL`, `CGU_DIV.S1SEL`, or `CGU_DIV.DSEL`) while the PLL is locked, but still aligning the clocks. This generates a CGU Error. If this error occurs it should be cleared and the desired values should be written to `CGU_DIV` again.

The `CGU_STAT.WDFMSERR` indicates a write access to the `CGU_CTL` register to change `CGU_CTL.DF` or `CGU_CTL.MSEL` while the PLL is locking. This generates a CGU Error. If this error occurs wait until the PLL has finished locking, clear the error, and write the desired `CGU_CTL.DF` or `CGU_CTL.MSEL` change again.

The `CGU_STAT.DIVERR` indicates a clock divisor value error, occurring when a CCLK divisor is greater than the SYSCLK divisor, as in: `CGU_DIV.CSEL > CGU_DIV.SYSSEL`. The CGU issues a CGU Error for this condition. If this error occurs it should be cleared and the new values should be written to `CGU_DIV` such that `CGU_DIV.CSEL` is less than or equal to `CGU_DIV.SYSSEL`.

## **CGU Generated Bus Errors**

The CGU will generate a bus error if a read or write transaction is attempted to an unused address within the CGU address range or if a misaligned access is made to an existing CGU register. In addition to the bus error `CGU_STAT.ADDRERR` will be set. If a write to a write protected DPM register is attempted a bus error will also be generated. In addition, `CGU_STAT.LWERR` will be set.

## **CGU Programming Model**

This section describes the programming concepts and mode configuration techniques for the CGU.

## **CGU Mode Configuration**

This section provides procedures related to clock and PLL configuration.

## Changing the PLLCLK Frequency

To change the PLLCLK frequency, write new values to `CGU_CTL.MSEL` or `CGU_CTL.DF`. Any time the PLL relocks, all core and system clocks are aligned.

1. Read `CGU_STAT` register. Verify that:
  - a. `PLLEN` bit = 1 (PLL enabled)
  - b. `PLLLK` bit = 1 (PLL is not locking) or `PLLLKERR` bit = 1 (PLL lock error, the PLL failed to lock)
  - c. `CLKSALGN` = 0 (clocks aligned)
2. Write the desired `CSEL`, `SOSEL`, `SYSSEL`, `S1SEL`, `DSEL` and `OSEL` values to the `CGU_DIV` register with the `UPDT` bit = 0.
3. Write the desired `DF` and `MSEL` values to the `CGU_CTL` register.
  - a. To change the PLL frequency while the cores are idle, write to the `CGU_CTL` register with the `WIDLE` bit set.
  - b. To change the PLL frequency while the cores are active, write to the `CGU_CTL` register with the `WIDLE` bit cleared.

*AFTER COMPLETING THIS TASK:*

This sequence updates the corresponding CGU registers, bypasses the PLL, makes the PLL lock to the new `MSEL` or `DF` values, changes the `CCLKn`, `SYSCLK`, `SCLKn`, `DCLK`, and `OCLK` frequencies and exits PLL bypass with all clocks aligned. When exiting the PLL bypass state, a CGU Event occurs.

The `CGU0_STAT` register exits this sequence with the `PLLEN` and `PLLLK` bits set, and the `PLLBP` and `CLKSALGN` bits cleared. The `PLLLK`, `PLLBP` and `CLKSALGN` bits can be polled to discover when the PLL is locked and the clocks are aligned.

Changing the PLL's frequency is allowed while the processor is in active (PLL bypassed) mode but the effect of the change is visible only after the transition to full-on mode. While in active mode a CGU Event occurs, indicating that the PLL has locked.

Changing the PLL frequency causes the `DCLK` frequency to change, therefore accessing the DDR memory, or the DDR memory controller's registers while the PLL and clocks frequencies' change is in progress may have unpredictable results.

## Changing the CCLKn, SYSCLK, or SCLKn frequency Without Modifying the PLLCLK Frequency

To change the clock frequencies is done by writing new `CGU_DIV.CnSEL`, `CGU_DIV.SYSSEL`, or `CGU_DIV.SnSEL` values. The frequency change occurs only when the PLL is not bypassed. Any time the

CCLKn, SYSCLK, or SCLKn clock frequencies is changed, they all exit the frequency change sequence aligned. The CGU\_SYSDCLK\_ALGN is not asserted even if the SYSCLK and the DCLK frequencies are equal.

1. Read the CGU\_STAT register. Verify that CLKSALGN = 0 (clocks aligned)
2. Write the desired CSEL, SOSEL, SYSSEL, S1SEL, DSEL and OSEL values to the CGU0\_DIV registers with the UPDT bit = 1.

*ADDITIONAL INFORMATION:* This write updates the CGU\_DIV register and changes the CCLKn, SYSCLK or SCLKn frequencies and aligns these clocks. When the clocks are aligned a CGU Event occurs.

*AFTER COMPLETING THIS TASK:*

The CGU\_STAT register exits this sequence with the CLKSALGN bit cleared. The CLKSALGN bit can be polled to discover when the clocks are aligned. Any write to the CGU\_DIV register intended to change an xSEL field while the CLKSALGN bit = 1 (clocks alignment in progress) triggers a bus error and the CGU\_DIV register is not modified.

Programming the SYSCLK frequency to a higher value than CCLKn also triggers an MMR access bus error and the CGU\_DIV register is not modified.

Writing to the CGU\_DIV register is allowed while the processor is in active (PLL bypassed) mode but the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Accessing the DDR memory while changing the SYSCLK frequency is not supported and may have unpredictable results.

## Changing the DCLK Clock Frequency

To change the DCLK clock frequency write a new CGU\_DIV.DSEL value. The frequency change occurs only when the PLL is not bypassed. Any time DCLK clock frequency is changed, DCLK, CCLKn, SYSCLK and SCLKn clocks exit the frequency change sequence aligned. The CGU\_SYSDCLK\_ALGN bit = 1 if the SYSCLK and the DCLK frequencies are equal.

1. Read the CGU0\_STAT register. Verify that CLKSALGN = 0 (clocks aligned)
2. Write the desired DSEL value to the CGU0\_DIV register, with the UPDT bit = 1.

*ADDITIONAL INFORMATION:* This write updates the CGU\_DIV register, changes the DCLK frequency, and aligns all clocks except OUTCLK.

*AFTER COMPLETING THIS TASK:*

The CGU\_STAT register exits this sequence with the CLKSALGN bit = 0. The CLKSALGN bit can be polled to discover when the clocks are aligned. Any write to the CGU\_DIV register intended to change the DSEL field while CLKSALGN is = 1 (clocks alignment in progress) triggers an MMR access bus error and the CGU\_DIV register is not modified. When clocks are aligned a CGU event occurs.

Writing to `CGU0_DIV.DSEL` is allowed while the processor is in active (PLL bypassed) mode but the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

Accessing the DDR memory, or the DDR memory controller's registers while changing the `DCLK` frequency is not supported and may have unpredictable results.

## Changing the OUTCLK Frequency

To change the OUTCLK clock frequency, write a new `CGU_DIV.OSEL` value. Any time OUTCLK clock frequency is changed, the OUTCLK, CCLKn, SYCLK and SCLKn clocks exit the frequency change sequence aligned. The `CGU_SYSDCLK_ALGN` signal is not modified.

1. Read `CGU_STAT` register. Verify that `CLKSALGN = 0` (clocks aligned)
2. Write desired `OSEL` value with the `UPDT` bit 1 to the `CGU0_DIV` register.

*ADDITIONAL INFORMATION:* This write updates the `CGU_DIV` register, changes the `DCLK` frequency, and aligns all clocks except OUTCLK.

*AFTER COMPLETING THIS TASK:*

The `CGU_STAT` register exits this sequence with the `CLKSALGN` bit = 0. The `CLKSALGN` bit can be polled to discover when the clocks are aligned. Any write to the `CGU_DIV` register intended to change the `DSEL` field while `CLKSALGN = 1` (clock alignment in progress) triggers an MMR access bus error and the `CGU_DIV` register is not modified. When clocks are aligned a CGU event occurs.

Writing to the `OSEL` field in the `CGU_DIV` register is allowed while the processor is in active (PLL bypassed) mode but the effect of the write is visible only after the transition to full-on (PLL not bypassed) mode.

## Aligning All Clocks

To align CCLKn, SYCLK, SCLKn, DCLK and OUTCLK clocks write 1 to `CGU0_DIV.ALGN`. The `CSEL`, `SYSEL`, `SnSEL`, `DSEL` or `OSEL` may be changed if a frequency change is also required. The `CGU_SYSDCLK_ALGN` is asserted if SYCLK and DCLK frequencies are equal.

1. Read the `CGU_STAT` register. Verify that `CLKSALGN = 0` (clocks aligned).
2. Write 1 to the `ALGN` bit in `CGU_DIV` register. All other fields may or may not change.

*ADDITIONAL INFORMATION:* This write does not alter the `CGU_DIV` register unless any of the `xSEL` fields is modified. When clocks are aligned a CGU event occurs.

*AFTER COMPLETING THIS TASK:*

The `CGU_STAT` register exits this sequence with the `CLKSALGN` bit = 0. The `CLKSALGN` bit can be polled to discover when the clocks are aligned. Any write to the `CGU0_DIV` register intended to align clocks or to

change an `xSEL` field while `CLKSALGN = 1` (clocks alignment in progress) triggers an MMR access bus error and the `CGU0_DIV` register is not modified.

Writing 1 to `CGU0_DIV.ALGN` has no effect while the processor is in active (PLL bypassed) mode. Accessing the DDR memory while changing the `DCLK` or `SYSCLK` frequencies is not supported and may have unpredictable results.

## Valid ADSP-BF60x `CGU_CTL.MSEL` Bit Settings

The valid range of `CGU_CTL.MSEL` settings is from 8 to 50 in decimal. All other settings are reserved. In addition to this requirement, `CGU_CTL.MSEL` must be set such that the minimum and maximum `PLLCLK` specified in the data sheet is not violated.

## CGU Register Descriptions

Clock Generation Unit (CGU) contains the following registers.

**Table 3-6:** CGU Module Registers

| Name                       | Description            |
|----------------------------|------------------------|
| <code>CGU_CTL</code>       | Control Register       |
| <code>CGU_STAT</code>      | Status Register        |
| <code>CGU_DIV</code>       | Divisor Register       |
| <code>CGU_CLKOUTSEL</code> | CLKOUT Select Register |

## Control Register

The `CGU_CTL` controls the clock generation divisors for `SYS_CLKIN` and the PLL. Read after write accesses to the `CGU_CTL` register returns the new value even if the clock's frequency change is still in progress.

**CGU\_CTL: Control Register - R/W**

Reset = 0x0000 1000

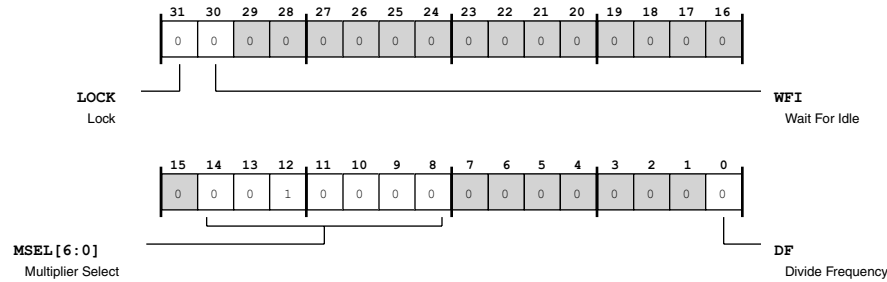


**Figure 3-3:** CGU\_CTL Register Diagram

**Table 3-7:** CGU\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the CGU_CTL.LOCK bit is set, the CGU_CTL register is read only (locked).                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 0      Unlock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1      Lock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 30      | WFI      | Wait For Idle.<br>Modifying the PLL multiplier requires the PLL to re-lock and once the PLL locks, clocks have to be synchronized. Changes to the CGU_CTL.MSEL and the CGU_CTL.DF result in bypassing the PLL. The CGU_CTL.WFI force the PLL to wait for all processor cores to be in an idle or reset state before changing frequencies as a result of change to the CGU_CTL.MSEL or CGU_CTL.DF fields. Write accesses to CGU_CTL to change CGU_CTL.DF or CGU_CTL.MSEL while the PLL is locking sets the CGU_STAT.WDFMSERR bit. |
|         |          | 0      Update Immediately                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Wait for Idle                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 3-7:** CGU\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14:8    | MSEL     | Multiplier Select.<br>The <code>CGU_CTL.MSEL</code> selects the multiplier in the PLLCLK equation:<br>$\text{PLLCLK frequency} = (\text{SYS\_CLKIN frequency} / (\text{DF} + 1)) * \text{MSEL}$<br>Where the value of MSEL may be between 1 and 127. Note that supported MSEL values are product specific. See the product specific section of the CGU for additional notes. |
|         |          | xxxxxxx MSEL = 1 to 127                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0 Reserved                                                                                                                                                                                                                                                                                                                                                                   |
| 0       | DF       | Divide Frequency.<br>The <code>CGU_CTL.DF</code> selects whether or not the SYS_CLKIN input is divided by two before being passed to the PLL.                                                                                                                                                                                                                                |
|         |          | 0 Pass OSC_CLKIN to PLL                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Pass OSC_CLKIN/2 to PLL                                                                                                                                                                                                                                                                                                                                                    |

## Status Register

The `CGU_STAT` register reflects the PLL status and errors detected during the PLL configuration. This register may be cleared asynchronously by a reset signal from the RCU module. All bits---except those defined as W1C (write-1-to-clear)---are read only.



**CGU\_STAT: Status Register - R/W**

Reset = 0x0000 03f5

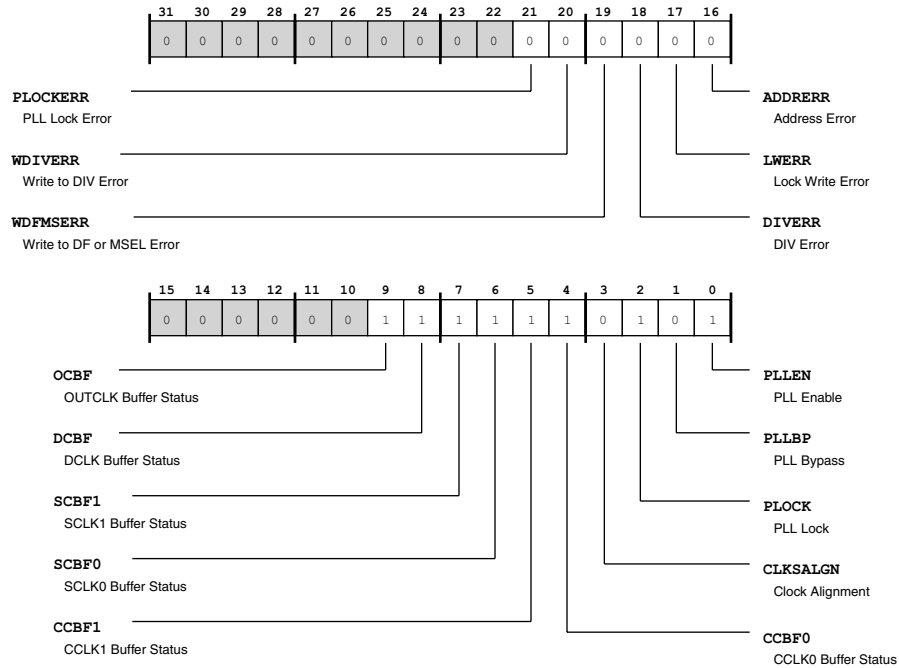


**Figure 3-4:** CGU\_STAT Register Diagram

**Table 3-8:** CGU\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                         |
|---------|----------|---------------------------------------------------------------------------------|
| 21      | PLOCKERR | PLL Lock Error.<br>The CGU_STAT.PLOCKERR indicates that the PLL failed to lock. |
|         |          | 0 No Error                                                                      |
|         |          | 1 PLL Lock Error                                                                |

**Table 3-8:** CGU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | WDIVERR  | Write to DIV Error.<br>The CGU_STAT.WDIVERR indicates a write access to the CGU_DIV register (to trigger an alignment sequence or to change CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.S0SEL, CGU_DIV.S1SEL, or CGU_DIV.DSEL) while the PLL is locked, but still aligning the clocks. Read after write accesses to the CGU_STAT and CGU_DIV registers return the new value even if the clock frequency change is still in progress. |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Write DIV Error                                                                                                                                                                                                                                                                                                                                                                                                              |
| 19      | WDFMSERR | Write to DF or MSEL Error.<br>The CGU_STAT.WDFMSERR indicates a write access to the CGU_CTL register to change CGU_CTL.DF or CGU_CTL.MSEL while the PLL is locking.                                                                                                                                                                                                                                                            |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Write DF/MSEL Error                                                                                                                                                                                                                                                                                                                                                                                                          |
| 18      | DIVERR   | DIV Error.<br>The CGU_STAT.DIVERR indicates a clock divisor value error, occurring when the CCLK clock divisor is greater than the SYSCLK clock divisor, as in:<br>$\text{CGU\_DIV.CSEL} > \text{CGU\_DIV.SYSSEL}$<br>The CGU issues a CGU error for this condition.                                                                                                                                                           |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 DIV Error                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 17      | LWERR    | Lock Write Error.<br>The CGU_STAT.LWERR indicates an attempt to write to write-protected (locked) CGU registers. The CGU issues a bus error for this condition.                                                                                                                                                                                                                                                                |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Lock Write Error                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 3-8:** CGU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | ADDRERR  | Address Error.<br>The <code>CGU_STAT.ADDRERR</code> indicates an attempt to make a read or write access to unimplemented addresses or accesses are non-aligned. The CGU issues a bus error for this condition. |
|         |          | 0                      No Error                                                                                                                                                                                |
|         |          | 1                      Address Error                                                                                                                                                                           |
| 9       | OCBF     | OUTCLK Buffer Status.<br>The <code>CGU_STAT.OCBF</code> indicates whether the OUTCLK buffer is enabled.                                                                                                        |
|         |          | 0                      Disabled                                                                                                                                                                                |
|         |          | 1                      Enabled                                                                                                                                                                                 |
| 8       | DCBF     | DCLK Buffer Status.<br>The <code>CGU_STAT.DCBF</code> indicates whether the DCLK buffer is enabled.                                                                                                            |
|         |          | 0                      Disabled                                                                                                                                                                                |
|         |          | 1                      Enabled                                                                                                                                                                                 |
| 7       | SCBF1    | SCLK1 Buffer Status.<br>The <code>CGU_STAT.SCBF1</code> indicates whether the SCLK1 buffer is enabled.                                                                                                         |
|         |          | 0                      Disabled                                                                                                                                                                                |
|         |          | 1                      Enabled                                                                                                                                                                                 |
| 6       | SCBF0    | SCLK0 Buffer Status.<br>The <code>CGU_STAT.SCBF0</code> indicates whether the SCLK0 buffer is enabled.                                                                                                         |
|         |          | 0                      Disabled                                                                                                                                                                                |
|         |          | 1                      Enabled                                                                                                                                                                                 |

**Table 3-8:** CGU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | CCBF1    | CCLK1 Buffer Status.<br>The <code>CGU_STAT.CCBF1</code> indicates whether the CCLK1 buffer is enabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Enabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 4       | CCBF0    | CCLK0 Buffer Status.<br>The <code>CGU_STAT.CCBF0</code> indicates whether the CCLK0 buffer is enabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Enabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 3       | CLKSALGN | Clock Alignment.<br>The <code>CGU_STAT.CLKSALGN</code> indicates whether a clock alignment sequence is in progress. This bit is set when clocks alignment is required by changes to <code>CGU_DIV.CSEL</code> , <code>CGU_DIV.S0SEL</code> , <code>CGU_DIV.S1SEL</code> , <code>CGU_DIV.DSEL</code> , or <code>CGU_DIV.OSEL</code> . The <code>CGU_STAT.CLKSALGN</code> bit is cleared when clocks are aligned. Note that (after a PLL frequency change in active state) the <code>CGU_STAT.CLKSALGN</code> bit may indicate that clocks are not aligned even though the clocks are aligned (all clocks are aligned and running at CLKIN's frequency). |
|         |          | 0 Clocks are Aligned                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Clocks not Aligned (alignment in progress)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 2       | PLOCK    | PLL Lock.<br>The <code>CGU_STAT.PLOCK</code> indicates whether the PLL is locked. This bit is set when the PLL locks (PLL lock counter end-of-count). The <code>CGU_STAT.PLOCK</code> bit is cleared when requested PLL frequency change (for PLL reset, PLL disable-to-enable transition, or change to <code>CGU_CTL.MSEL</code> or <code>CGU_CTL.DF</code> ) is in progress.                                                                                                                                                                                                                                                                         |
|         |          | 0 PLL not Locked (PLL frequency change in progress)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 PLL Locked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Table 3-8:** CGU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | PLLBP    | PLL Bypass.<br>The <code>CGU_STAT.PLLBP</code> indicates whether the PLL is bypassed. The default value for <code>CGU_STAT.PLLBP</code> is determined by the bypass strap pin. |
|         |          | 0 PLL not Bypassed                                                                                                                                                             |
|         |          | 1 PLL Bypassed                                                                                                                                                                 |
| 0       | PLLEN    | PLL Enable.<br>The <code>CGU_STAT.PLLEN</code> indicates whether the PLL is enabled.                                                                                           |
|         |          | 0 Disabled                                                                                                                                                                     |
|         |          | 1 Enabled                                                                                                                                                                      |

## Divisor Register

The `CGU_DIV` register controls clock divisors for core clocks, system clocks, external (off core) memory clocks, and output clock. Read after write accesses to the `CGU_DIV` register returns the new value even if the clock's frequency change is still in progress.

### CGU\_DIV: Divisor Register - R/W

Reset = 0x0408 4844

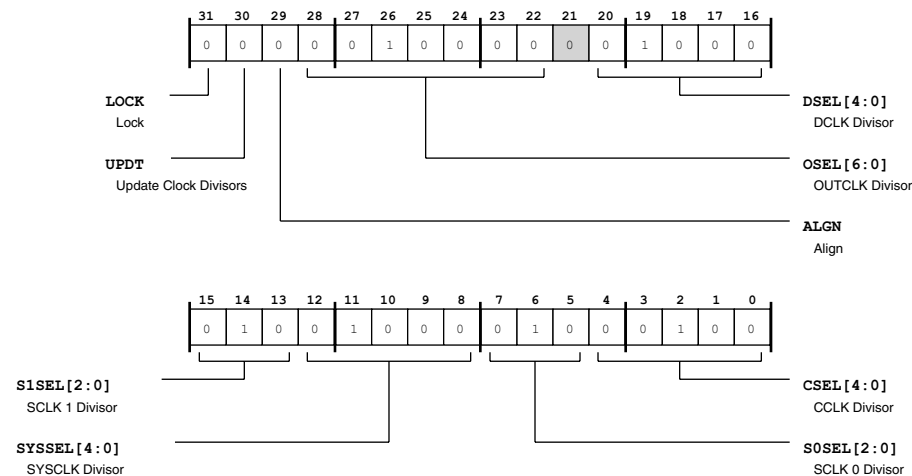


**Figure 3-5:** CGU\_DIV Register Diagram

**Table 3-9:** CGU\_DIV Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the CGU_DIV.LOCK bit is set, the CGU_DIV register is read only (locked).                                                                               |
|         |          | 0                      Unlock                                                                                                                                                                                           |
|         |          | 1                      Lock                                                                                                                                                                                             |
| 30      | UPDT     | Update Clock Divisors.<br>The CGU_DIV.UPDT controls whether the CGU drives new CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.SOSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and CGU_DIV.OSEL values to PLL after CGU_DIV register update.  |
|         |          | 0                      No PLL Update                                                                                                                                                                                    |
|         |          | 1                      Drive Updated SEL Values to PLL                                                                                                                                                                  |
| 29      | ALGN     | Align.<br>The CGU_DIV.ALGN directs the CGU to align the PLL-based clocks. The divisor selections (CGU_DIV.CSEL, CGU_DIV.SYSSEL, CGU_DIV.SOSEL, CGU_DIV.S1SEL, CGU_DIV.DSEL, and/or CGU_DIV.OSEL) do not have to change. |
|         |          | 0                      No Action                                                                                                                                                                                        |
|         |          | 1                      Align PLL Clocks                                                                                                                                                                                 |
| 28:22   | OSEL     | OUTCLK Divisor.<br>The CGU_DIV.OSEL selects the divisor in the OUTCLK equation:<br>OUTCLK frequency = (SYS_CLKIN frequency / (DF+1)) * MSEL / CGU_DIV.OSEL<br>Where the value of CGU_DIV.OSEL is between 1 and 127.     |
|         |          | xxxxxxx              OSEL = 1 to 127                                                                                                                                                                                    |
|         |          | 0                      Reserved                                                                                                                                                                                         |

**Table 3-9:** CGU\_DIV Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 20:16   | DSEL     | DCLK Divisor.<br>The <code>CGU_DIV.DSEL</code> selects the divisor in the DCLK equation:<br>$\text{DCLK frequency} = (\text{SYS\_CLKIN frequency} / (\text{DF}+1)) * \text{MSEL} / \text{CGU\_DIV.DSEL}$ Where the value of <code>CGU_DIV.DSEL</code> is between 1 and 31.             |                  |
|         |          | 0                                                                                                                                                                                                                                                                                      | Reserved         |
|         |          | xxxxx                                                                                                                                                                                                                                                                                  | DSEL = 1 to 31   |
| 15:13   | S1SEL    | SCLK 1 Divisor.<br>The <code>CGU_DIV.S1SEL</code> selects the divisor in the SCLK1 equation:<br>$\text{SCLK1 frequency} = (\text{SYSCLK frequency}) / \text{CGU\_DIV.S1SEL}$ Where the value of <code>CGU_DIV.S1SEL</code> is between 1 and 7.                                         |                  |
|         |          | 0                                                                                                                                                                                                                                                                                      | Reserved         |
|         |          | xxx                                                                                                                                                                                                                                                                                    | S1SEL = 1 to 7   |
| 12:8    | SYSSEL   | SYSCLK Divisor.<br>The <code>CGU_DIV.SYSSEL</code> selects the divisor in the SYSCLK equation:<br>$\text{SYSCLK frequency} = (\text{SYS\_CLKIN frequency} / (\text{DF}+1)) * \text{MSEL} / \text{CGU\_DIV.SYSSEL}$ Where the value of <code>CGU_DIV.SYSSEL</code> is between 1 and 31. |                  |
|         |          | 0                                                                                                                                                                                                                                                                                      | Reserved         |
|         |          | xxxxx                                                                                                                                                                                                                                                                                  | SYSSEL = 1 to 31 |
| 7:5     | S0SEL    | SCLK 0 Divisor.<br>The <code>CGU_DIV.S0SEL</code> selects the divisor in the SCLK0 equation:<br>$\text{SCLK0 frequency} = (\text{SYSCLK frequency}) / \text{CGU\_DIV.S0SEL}$ Where the value of <code>CGU_DIV.S0SEL</code> is between 1 and 7.                                         |                  |
|         |          | 0                                                                                                                                                                                                                                                                                      | Reserved         |
|         |          | xxx                                                                                                                                                                                                                                                                                    | S0SEL = 1 to 7   |

**Table 3-9:** CGU\_DIV Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4:0     | CSEL     | CCLK Divisor.<br>The CGU_DIV.CSEL selects the divisor in the CCLK equation:<br>$CCLK \text{ frequency} = (\text{SYS\_CLKIN frequency} / (DF+1)) * MSEL / CGU\_DIV.CSEL$<br>Where the value of CGU_DIV.CSEL is between 1 and 31. |
|         |          | 0 Reserved                                                                                                                                                                                                                      |
|         |          | xxxxx CSEL= 1 to 31                                                                                                                                                                                                             |

## CLKOUT Select Register

The CGU\_CLKOUTSEL selects the signal that the CGU drives through the CLKOUT multiplexer.

**CGU\_CLKOUTSEL: CLKOUT Select Register - R/W**

Reset = 0x0000 0000

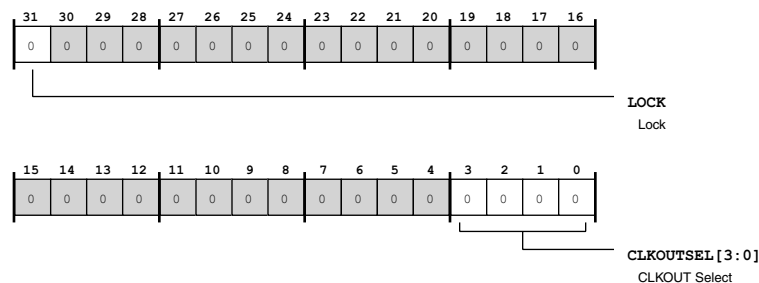


**Figure 3-6:** CGU\_CLKOUTSEL Register Diagram

**Table 3-10:** CGU\_CLKOUTSEL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the CGU_CLKOUTSEL.LOCK bit is set, the CGU_CLKOUTSEL register is read only (locked). |
|         |          | 0 Unlock                                                                                                                                              |
|         |          | 1 Lock                                                                                                                                                |



**Table 3-10:** CGU\_CLKOUTSEL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                               |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | CLKOUTSEL | CLKOUT Select.<br>The <code>CGU_CLKOUTSEL.CLKOUTSEL</code> selects the signal that the CGU drives through the CLKOUT pin multiplexer. |
|         |           | 0 CLKIN                                                                                                                               |
|         |           | 1 CCLKn/4                                                                                                                             |
|         |           | 2 SYSCLK/2                                                                                                                            |
|         |           | 3 SCLK0                                                                                                                               |
|         |           | 4 SCLK1                                                                                                                               |
|         |           | 5 DCLK/2                                                                                                                              |
|         |           | 6 Reserved                                                                                                                            |
|         |           | 7 OUTCLK                                                                                                                              |
|         |           | 8 Reserved                                                                                                                            |
|         |           | 9 Reserved                                                                                                                            |
|         |           | 10 Reserved                                                                                                                           |
|         |           | 11 GND (Disable OUTCLK)                                                                                                               |
|         |           | 11xx Reserved                                                                                                                         |



## 4 System Protection Unit (SPU)

The system protection unit (SPU) provides features that let you protect system resources from errant writes. A number of protection categories (types of registers to protect) are available.

### SPU Features

The System Protection Unit includes the following abilities.

- To write-protect system MMR from certain system masters.
- To simultaneously lock multiple peripheral configuration registers.
- To write-protect and block access to its own write-protection registers from other system masters.

### SPU Functional Description

The SPU has a register associated with each peripheral. Each of these write-protection registers has the exact same bits that correspond to a particular SMMR master (Core 0, Core 1, MDMA, for example). When the bits are set, the corresponding SMMR masters are locked out of accessing the associated peripheral's register address space. The bits in the register can be cleared to allow access to the peripheral's registers again. Any writes that are in progress when write-protection is initiated will complete before subsequent writes are blocked.

In the following figure, each write-protect register in the SPU is associated with a particular peripheral.

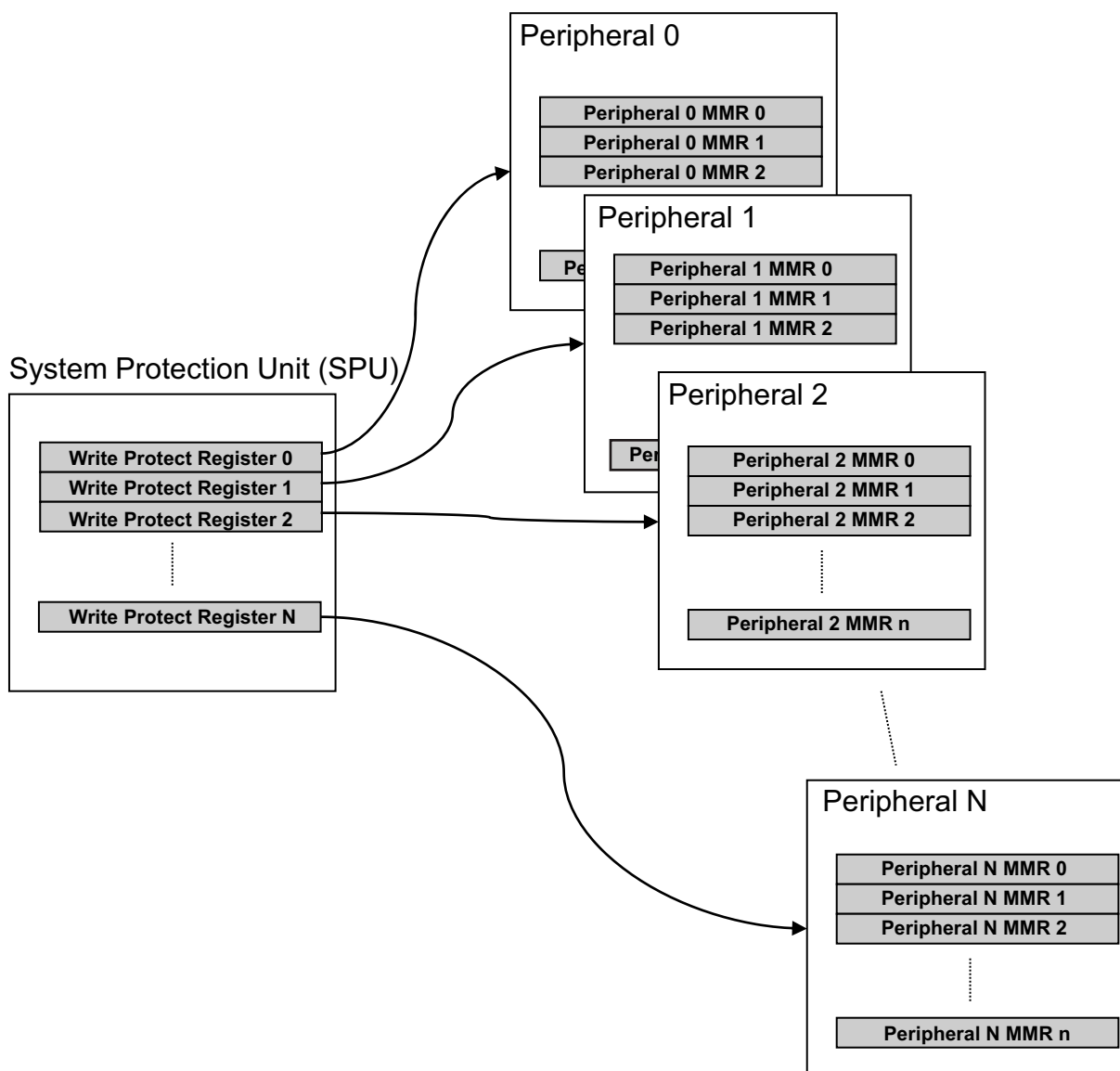
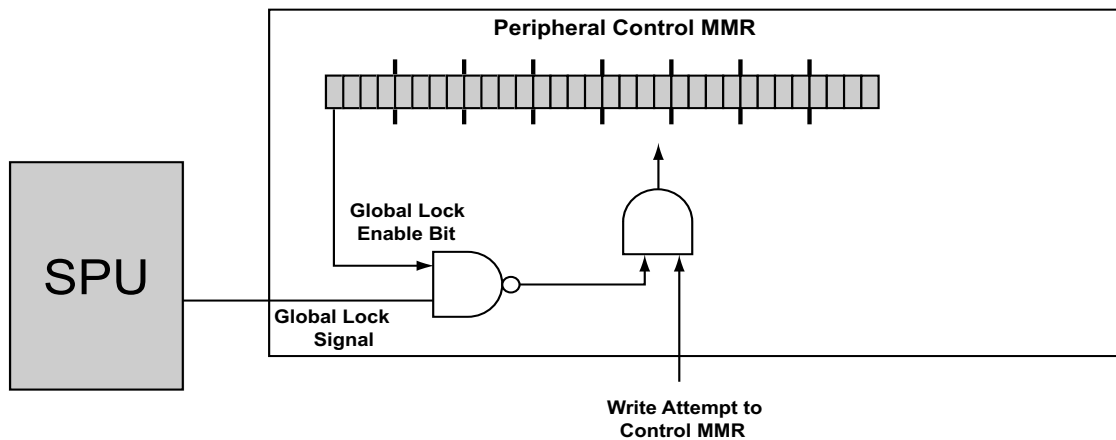


**Figure 4-1:** SPU Write Protect Registers

The SPU also has global locking capability. When enabled, a system-wide global lock signal is active. Some peripherals have a lock enable bit in their control register. When this bit is set, the peripheral recognizes the global lock signal and blocks further write-accesses to its own control register. Access to the peripheral's configuration register is re-enabled when global lock is turned off in the SPU.

The following figure is a conceptual diagram where a peripheral will block any write attempts to its control register if the global lock signal from the SPU is active AND the global lock enable bit is set in the peripheral's control register.



### Figure 4-2: Global Locking

The SPU can write protect its own registers. When the write protection register lock bit is set and global locking is enabled, accesses to the SPU write-protection registers are blocked. To re-enable write access to the write-protection registers in the SPU, global locking must be disabled.

In the following figure a write-protect register in the SPU blocks write-attempts to the associated peripheral's MMR space. The bits in the write-protect register specify which masters to block write-access from.

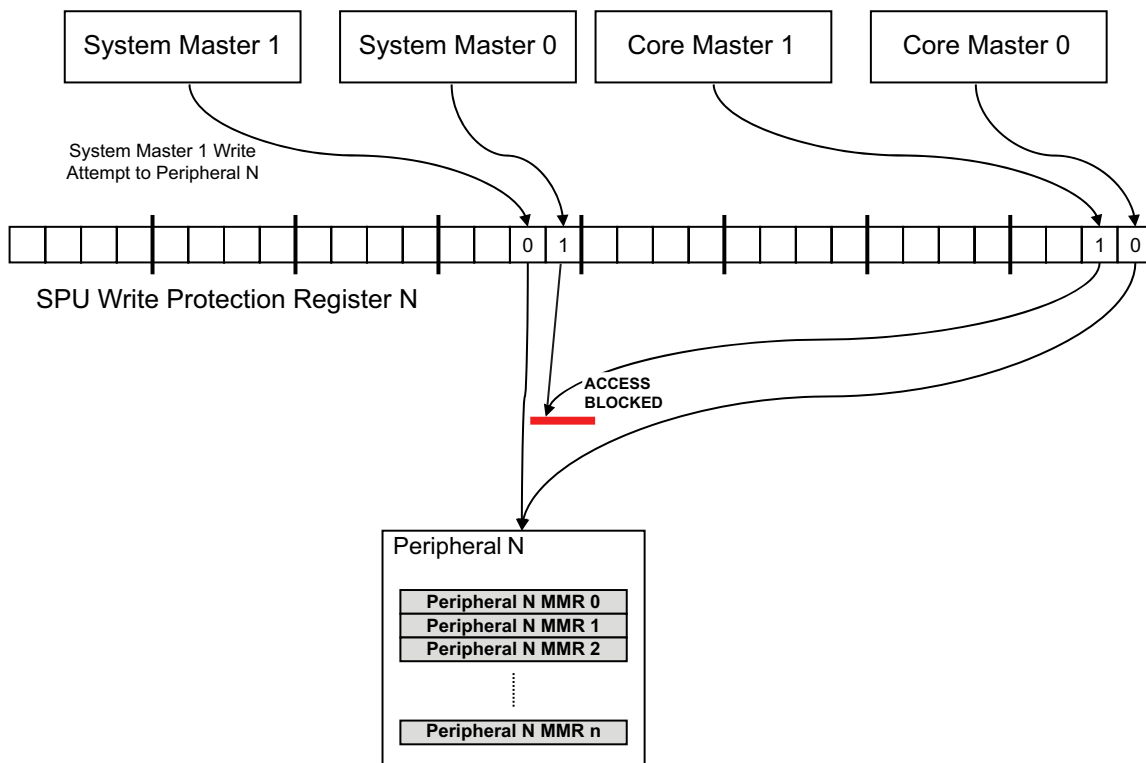


**Figure 4-3:** SPU Write-Protect Register Blocking Access from System Master 0 and Core Master 1

## SPU Register List

The system protection unit (SPU) provides a set of registers that allow you to protect system resources from errant writes. The protection categories are global lock (protects configuration registers) and write protect register lock (protects the write protect register). For more information on SPU functionality, see the SPU register descriptions.

**Table 4-1:** SPU Module Registers

| Name     | Description              |
|----------|--------------------------|
| SPU_CTL  | Control Register         |
| SPU_STAT | Status Register          |
| SPU_WPn  | Write Protect Register n |

## SPU Definitions

### Write-Protect Register

Memory mapped registers in the SPU. Each register correlates to a specific peripheral instance. It controls the write access to the peripheral's register set.

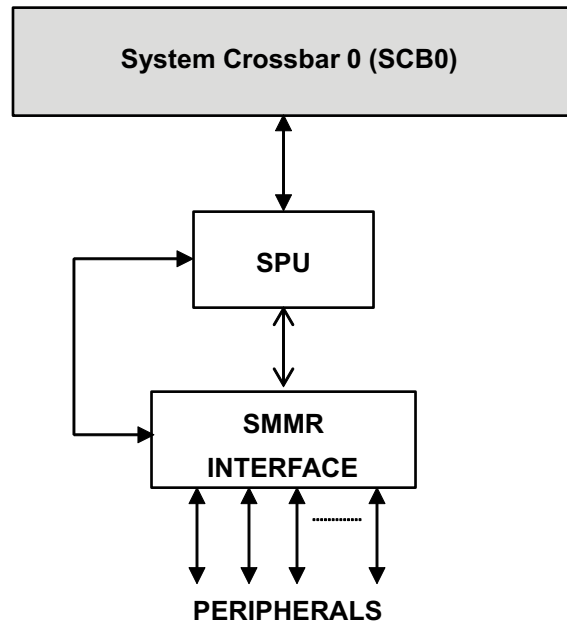
### Global Locking

SPU's ability to prevent write access to multiple peripheral's control register at once.

## SPU Block Diagram

The figure below shows a system level block diagram of where the SPU is in the system. It sits in between the SMMR interface and the system crossbar. Depending on the configuration of the SPU write-protect registers, it can block access to certain peripherals from certain SMMR masters.

SPU System Level Block Diagram



## SPU Architectural Concepts

As shown in the block diagram, the SPU sits between the System Crossbar (SCB) and the SMMR interface to the peripherals. Any MMR access to any peripheral from any master comes through the SCB and is gated by the SPU. Depending on the configuration of the write-protection registers in the SPU, the SPU may or may not allow the MMR write to go through.

## SPU Event Control

The system protection unit provides write protection against a peripheral's MMRs and its own write-protect registers. If a write attempt is made to any peripheral's MMR and was locked, the SPU will block the write and generate a bus error to the master that attempted the write. That master may or may not generate an event based upon the returned error. The SPU does not generate an event for blocked write attempts.

The SPU can also lock its own registers from write attempts. If a write-attempt was made to a locked register in the SPU, the SPU blocks it and records it as an error in `SPU_STAT.LWERR`. Again, the SPU generates a bus error to the master that attempted the write. The master may or may not generate an event based upon the returned error. The SPU does not generate an event for a blocked write access to an SPU register.

## SPU Programming Model

The system protection unit (SPU) consists of write-protect registers. Each one corresponds to a different peripheral instance. Bits in the write-protect registers correspond to system masters that can modify the

MMR contents of the peripherals. By writing to these write-protect registers, the corresponding peripheral's memory mapped registers will be write protected against masters whose bits in the write-protect register have been set.

Another capability of the SPU is to globally lock peripherals' control register. Peripherals that support this feature will have a lock enable bit in their control register. When the global lock signal is active from the SPU and the peripheral's lock enable bit is set, the peripheral blocks any more write attempts to its control register from any master. If the lock enable bit of a peripheral is not set and the global lock signal is active, access to that peripheral's control register is still allowed. To grant access again, the global lock signal from the SPU must be disabled. This is done by writing the value 0xAD into the GLCK field of the SPU\_CTL register.

Another protection mechanism that the SPU offers is write protection against the write-protection registers. If the Write Protect Register Lock (WPLCK) bit is set in the SPU\_CTL register and the global lock signal is active, writes to the SPU's write-protect registers will be blocked. To re-enable access to the write-protect registers in the SPU, the global lock signal must be deactivated by writing 0xAD into the GLCK field of the SPU\_CTL register.

## **SPU Mode Configuration**

The SPU can provide address range wide protection by write-protecting the peripherals MMR address range from system MMR masters. It can also provide register wide protection by using Global Locking. Peripherals that support this feature can enable it their respective configuration register. When the SPU enables the Global Lock signal, all subsequent writes to the peripheral's configuration register are blocked until the Global Lock signal is deasserted. Similarly, the SPU's own write-protection registers can be write protected using the Global Lock signal as well. All these modes of operation can be used in conjunction.

## **Locking Write-Protect Registers**

Use the following steps to lock (write protect) a register.

1. Set SPU\_CTL.WPLCK bit and set the SPU\_CTL.GLOCK field to something other than 0xAD.

*RESULT:*

The SPU write-protect registers are blocked from further write accesses.



## Protecting a Peripheral

Use the following procedure to protect a peripheral

1. Determine which peripheral needs protection and locate the corresponding write-protect register in the SPU.
2. Determine which SMMR master(s) the peripheral needs to be protected from and set the corresponding bit(s) in the write-protect register for the peripheral in the SPU.

### *RESULT:*

After setting the write-protect register for the particular peripheral, the SMMR master(s) will be blocked from writing to any MMR in the peripheral's address space until the bits in the write-protect register are cleared.

## SPU Register Descriptions

System Protection Unit (SPU) contains the following registers.

**Table 4-2:** SPU Module Registers

| Name     | Description              |
|----------|--------------------------|
| SPU_CTL  | Control Register         |
| SPU_STAT | Status Register          |
| SPU_WPn  | Write Protect Register n |

## Control Register

The SPU control register (SPU\_CTL) provides a global lock for configuration registers and write protection for registers.

SPU\_CTL: Control Register - R/W

Reset = 0x0000 00ad

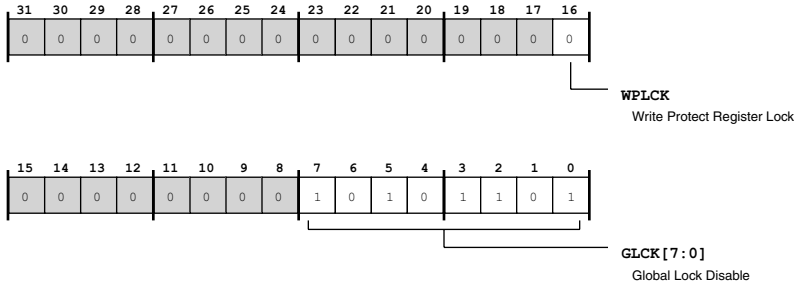


Figure 4-4: SPU\_CTL Register Diagram

Table 4-3: SPU\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | WPLCK    | Write Protect Register Lock.<br>The SPU_CTL.WPLCK works with the SPU_CTL.GLCK field. If the write protect register lock is enabled (SPU_CTL.WPLCK bit =1) and the global lock is enabled, writes to the SPU_WPn register are disabled (locked out). |
|         |          | 0      Disable                                                                                                                                                                                                                                      |
|         |          | 1      Enable                                                                                                                                                                                                                                       |
| 7:0     | GLCK     | Global Lock Disable.<br>The SPU_CTL.GLCK controls the global lock of configuration registers. Writing 0xAD to this field disables the lock, and writing any other value enables the lock.                                                           |

Status Register

The SPU\_STAT register indicates the error and lock status for the SPU.

**SPU\_STAT: Status Register - R/W**

Reset = 0x0000 0000

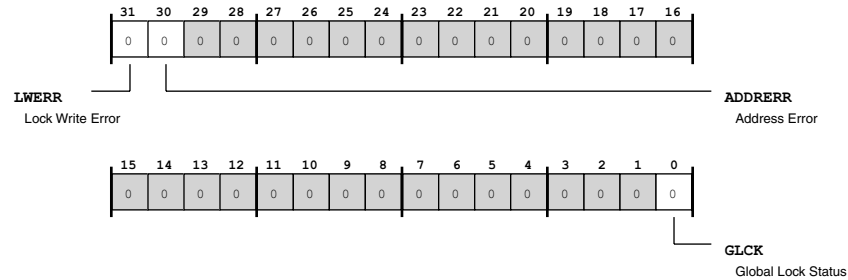


**Figure 4-5:** SPU\_STAT Register Diagram

**Table 4-4:** SPU\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LWERR    | Lock Write Error.<br>The SPU_STAT.LWERR is write-1-to-clear and indicates whether there has been an attempted write to a register with its LOCK bit set while SPU_CTL.GLCK was asserted. |
|         |          | 0 Inactive                                                                                                                                                                               |
|         |          | 1 Active                                                                                                                                                                                 |
| 30      | ADDRERR  | Address Error.<br>The SPU_STAT.ADDRERR is write-1-to-clear and indicates whether there has been an attempted write to a read-only register or an access an invalid address.              |
|         |          | 0 Inactive                                                                                                                                                                               |
|         |          | 1 Active                                                                                                                                                                                 |
| 0       | GLCK     | Global Lock Status.<br>The SPU_STAT.GLCK indicates whether the global lock is enabled or disabled.                                                                                       |
|         |          | 0 Disabled (global_lock=0)                                                                                                                                                               |
|         |          | 1 Enabled (global_lock=1)                                                                                                                                                                |

Write Protect Register n

In the system, each SPU\_WPn register is assigned to a specific MMR address range associated with one peripheral. When the appropriate bits are set, writes to the peripheral from a specific master are blocked and an error is returned to the master. For more information, see the SPU functional description.

SPU\_WPn: Write Protect Register n - R/W

Reset = 0x0000 0000

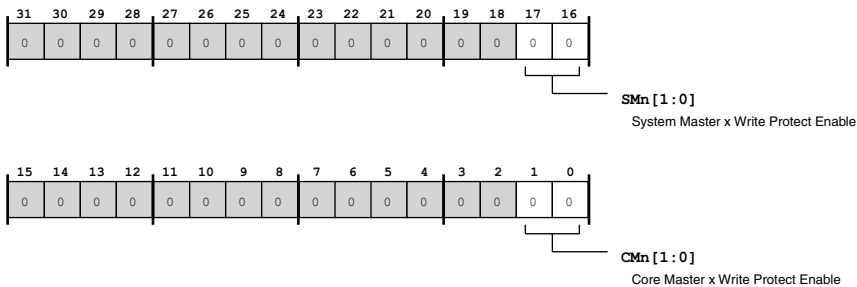


Figure 4-6: SPU\_WPn Register Diagram

Table 4-5: SPU\_WPn Register Fields

| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 17:16   | SMn      | System Master x Write Protect Enable. |
| 1:0     | CMn      | Core Master x Write Protect Enable.   |

# 5 Dynamic Power Management (DPM)

The dynamic power management (DPM) unit of the processor controls transitions between different power saving modes. The DPM also allows individual clock domains to be enabled and disabled.

## DPM Features

The DPM allows programs to control the processor's power mode as follows.

- The capability to shut off individual clock domains to save power
- The capability to bypass the PLL for power savings
- HIBERNATE mode allows `VDD_INT` to be shut off in order to save power
- Multiple external wake-up sources

## DPM Functional Description

The processor supports four different power domains, which maximizes flexibility while maintaining compliance with industry standards and conventions. By isolating the internal logic of the processor into its own power domain, separate from other I/O, the processor can take advantage of dynamic power management without affecting the other I/O devices. There are no sequencing requirements for the various power domains, but all domains must be powered according to the appropriate specifications, even if the feature/peripheral is not used. For more information on power domains, see the *ADSP-BF60x Blackfin Dual Core Embedded Processor* data sheet.

The dynamic power management feature of the processor allows the processor's core clock frequency ( $f_{\text{CLK}}$ ) to be dynamically controlled.

## DPM Register List

The dynamic power management (DPM) unit includes the phase locked loop (PLL) enable/disable features, deep sleep and hibernate mode controls, and clock domain enable/disable features. The combination of these features and controls provide selective and flexible power management. A set of registers govern DPM operations. For more information on DPM functionality, see the DPM register descriptions.

**Table 5-1:** DPM Module Registers

| Name               | Description                              |
|--------------------|------------------------------------------|
| DPM_CTL            | Control Register                         |
| DPM_STAT           | Status Register                          |
| DPM_CCBF_DIS       | Core Clock Buffer Disable Register       |
| DPM_CCBF_EN        | Core Clock Buffer Enable Register        |
| DPM_CCBF_STAT      | Core Clock Buffer Status Register        |
| DPM_CCBF_STAT_STKY | Core Clock Buffer Status Sticky Register |
| DPM_SCBF_DIS       | System Clock Buffer Disable Register     |
| DPM_WAKE_EN        | Wakeup Enable Register                   |
| DPM_WAKE_POL       | Wakeup Polarity Register                 |
| DPM_WAKE_STAT      | Wakeup Status Register                   |
| DPM_HIB_DIS        | Hibernate Disable Register               |
| DPM_PGCNTR         | Power Good Counter Register              |
| DPM_RESTOREn       | Restore n Register                       |

## ADSP-BF60x DPM Interrupt List

**Table 5-2:** ADSP-BF60x DPM Interrupt List

| Description | ID  | DMA Channel | Sensitivity |
|-------------|-----|-------------|-------------|
| DPM0 Event  | 131 |             | LEVEL       |

## DPM Definitions

To make the best use of the DPM, it is useful to understand the following terms.

### **Active**

Power saving mode in which the PLL is bypassed but still enabled.

### **Active with PLL disabled**

Power saving mode in which the PLL is bypassed and disabled.

### **CGU**

Clock Generation Unit which is comprised of the PLL and PCU

### **Deep Sleep**

Power saving mode in which all CCLKs are gated.

### **DPM**

Dynamic Power Management Controller

### **Full On**

The normal operating mode in which all clock domains are derived from the PLL.

### **Hibernate**

A power saving mode in which VDD\_INT can be shut off and the contents of on-chip memory isn't retained.

### **PCU**

PLL control unit

### **PLL**

Phase locked loop

### **RCU**

Reset control unit

## DPM Operating Modes

The DPM includes several operating modes described below. They are RESET, FULL-ON, ACTIVE, ACTIVE with PLL Disabled, DEEP SLEEP, and HIBERNATE.

## RESET

RESET is the initial state of the processor and is the result of a hardware or software triggered event. Entering RESET is not triggered by the DPM itself, but by the external `SYS_HWRST` pin or by the RCU. The DPM responds, as other functional units do, by transitioning to its default state with the exception of certain registers (described under HIBERNATE below) that are preserved if returning from hibernate. From RESET the DPM always transitions to FULL-ON.

## FULL-ON

FULL-ON is the DPM's default state after RESET. In this mode the processor can reach its maximum clock rate and power dissipation can be at its highest. The DPM transitions to ACTIVE if `DPM_CTL.PLLBPST` is set, to DEEP SLEEP if `DPM_CTL.DEEPSLEEP` is set, and to HIBERNATE if `DPM_CTL.HIBERNATE` is set.

## ACTIVE

In ACTIVE power is reduced on the `VDD_INT` domain (compared to FULL-ON) by bypassing the PLL and running clock domains at the `SYS_CLKIN` frequency. The processor is fully functional. The DPM transitions to FULL ON if `DPM_CTL.PLLBPCL` is set, ACTIVE with PLL disabled if `DPM_CTL.PLLDIS` is set, to DEEP SLEEP if `DPM_CTL.DEEPSLEEP` is set, and to HIBERNATE if `DPM_CTL.HIBERNATE` is set.

## ACTIVE with PLL Disabled

In ACTIVE with PLL Disabled power is reduced on the `VDD_INT` domain (compared to ACTIVE) by disabling the PLL in addition to running all units at the `SYS_CLKIN` frequency. The processor is fully functional. The DPM transitions to ACTIVE if `DPM_CTL.PLLDIS` is cleared, to DEEP SLEEP if `DPM_CTL.DEEPSLEEP` is set, and to HIBERNATE if `DPM_CTL.HIBERNATE` is set.

## DEEP SLEEP

DEEP SLEEP is entered by setting `DPM_CTL.DEEPSLEEP` and having all cores in idle. It is the programs responsibility in software to guarantee that system transfers including DMA are stopped before going to idle in each core and entering DEEP SLEEP. In this state power on the `VDD_INT` domain is reduced (compared to ACTIVE or GATED ACTIVE) by gating all the core and system clocks, and disabling the PLL.

Enabled hardware wake-ups or a hardware reset can make the part exit DEEP SLEEP. The `WSn` bits in `DPM_WAKE_EN` and `DPM_WAKE_POL` registers determine which hardware wake ups are enabled and what their polarity is. Wake-up assertions are latched only if they are enabled. The enabled wake-up assertion that occurs first is recorded in `DPM_WAKE_STAT`. To see which wake-up sources your processor reports see the BF60x Wake-Up Sources topic.



When a wake-up occurs, the DPM signals a DPM event interrupt to the SEC, transitions to ACTIVE mode and enables all clocks domains that are not disabled in the `DPM_SCBF_DIS` register. The DPM event interrupt will stay active until the user clears any bits that are set in `DPM_WAKE_STAT`. The DPM event interrupt is the first indication that the processor has exited DEEP SLEEP. It can be enabled in the SEC to make a core exit idle and to enable the corresponding core clock buffer. An alternate option to wake up a core is to enable the CGU event interrupt which will assert after the PLL locks.

## HIBERNATE

In HIBERNATE there is no power dissipation on the `VDD_INT` domain as long as the voltage regulator powering `VDD_INT` is shut off. If state information, data, or code needs to be preserved it can be stored in the `DPM_RESTORE` registers or in external memory. If dynamic memory is used the memory device needs to be placed into self-refresh mode before going to hibernate.

HIBERNATE is entered by setting `DPM_CTL.HIBERNATE`. It is the programs responsibility in software to guarantee that system transfers including DMA are stopped before entering HIBERNATE.

Once `DPM_CTL.HIBERNATE` has been set, the DPM will preserve the state of `DPM_WAKE_EN`, `DPM_WAKE_POL`, `DPM_HIB_DIS`, `DPM_PGCNTR`, and all `DPM_RESTORE` registers in the DPM-HV which is powered by the `VDD_EXT` domain. All other registers (with the exception of those mentioned in the RCU chapter) and on-chip memory will be erased once `VDD_INT` is powered down. After the contents of the registers listed above are moved to storage on the `VDD_EXT` domain, the `SYS_EXTWAKE` signal is de-asserted to indicate that it is safe to shut off the regulator providing voltage to `VDD_INT`.

Enabled hardware wake-ups or a hardware reset can begin the process of exiting HIBERNATE. The `WSn` bits in `DPM_WAKE_EN` and `DPM_WAKE_POL` registers determine which hardware wake ups are enabled and what their polarity is. Wake-up assertions are latched only if they are enabled. The enabled wake-up assertion that occurs first is recorded in `DPM_WAKE_STAT`. After a wake-up occurrence is latched by the processor, the `SYS_EXTWAKE` signal will be asserted to indicate that the regulator supplying `VDD_INT` should begin to ramp up. The DPM will then wait for both the `SYS_PWRGD` signal to be asserted and for the `DPM_PGCNTR.CNT` counter to expire which indicate that `VDD_INT` has been restored within the operating conditions specified in the data sheet. For more information about the Power Good Counter and `SYS_PWRGD` signal see [Ensuring VDD\\_INT is Restored Before Booting](#).

Once `VDD_INT` is restored, `SYS_PWRGD` is asserted, and `DPM_PGCNTR.CNT` is expired, the processor exits HIBERNATE and goes to RESET and the PLL begins locking. While the PLL is locking `DPM_WAKE_EN`, `DPM_WAKE_POL`, `DPM_HIB_DIS`, `DPM_PGCNTR`, and all `DPM_RESTORE` registers are restored from the DPM-HV. After coming out of RESET and locking the PLL the processor begins executing the boot code. Some processors support memory boot when returning from hibernate (instead of the boot mode selected by the `SYS_BMODE` signals). For more information see the booting chapter.

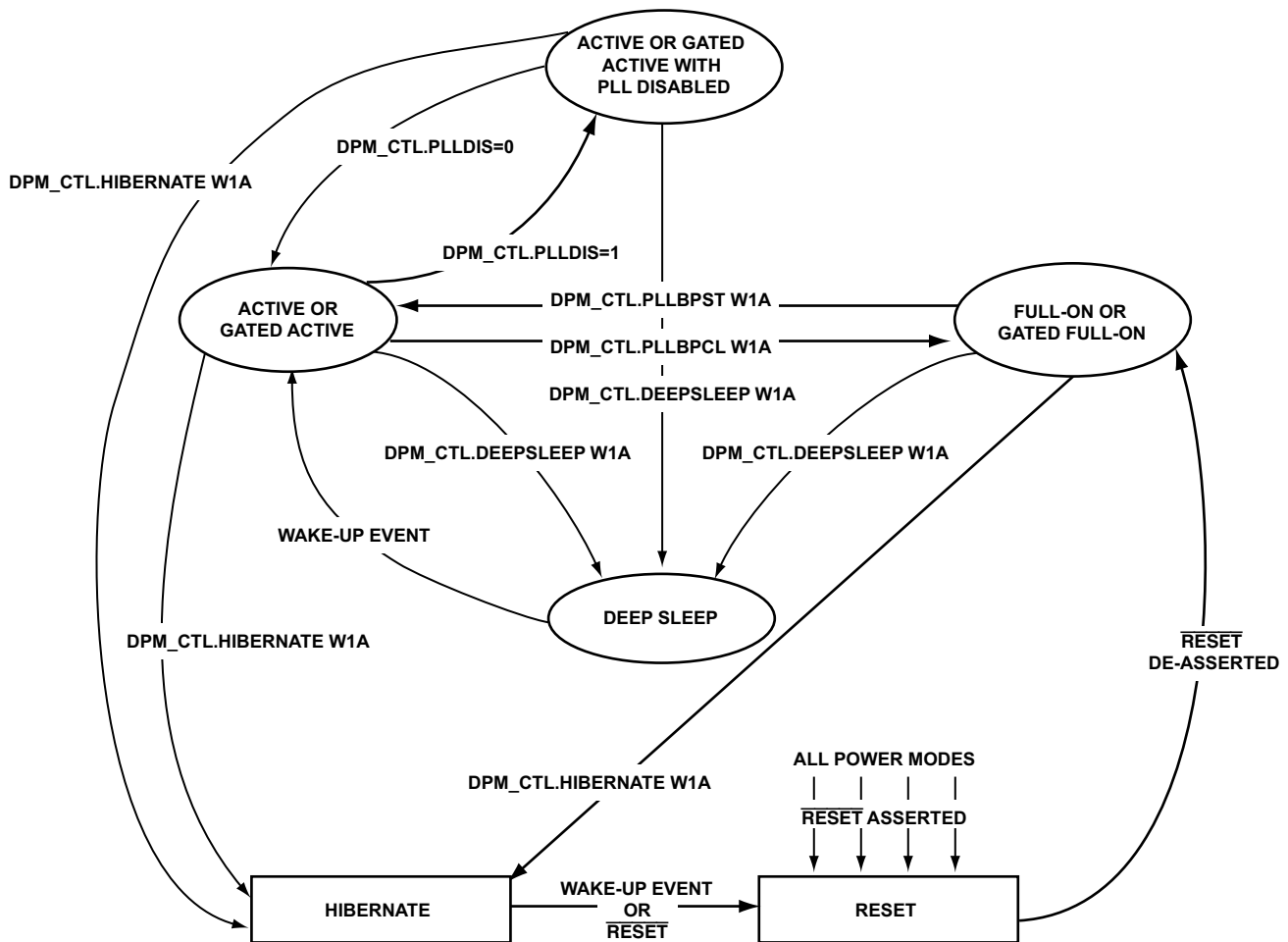


**Figure 5-1: Operating Modes and Transitions**

## Ensuring VDD\_INT is Restored Before Booting

Before the processor boots after returning from HIBERNATE VDD\_INT must be restored. This topic discusses how to signal to the processor that VDD\_INT has been restored.

After a wake-up occurrence is latched by the DPM while in HIBERNATE, the DPM power good counter will begin decrementing and the SYS\_EXTWAKE signal will be asserted to indicate to the regulator supplying VDD\_INT that it should begin to ramp up. It is the user's responsibility to then signal the DPM through software (the DPM power good counter in the DPM\_PGCNTR register) or hardware (the SYS\_PWRGD signal) that it is safe to begin the boot process. Note that the DPM will not kick off the boot process until SYS\_PWRGD is asserted and the DPM\_PGCNTR.CNT counter is expired.

## Using the Power Good Counter to Ensure That Internal Power is Restored

In order to use the software approach, which utilizes the DPM power good counter in the DPM\_PGCNTR register, it is suggested that SYS\_PWRGD be pulled up to VDD\_EXT. The DPM will then have to wait only for the DPM power good counter to expire before kicking off the boot process. It is the user's responsibility to select the appropriate setting for DPM\_PGCNTR.CNT such that when the counter expires, VDD\_INT has been restored to within the operating conditions specified in the data sheet. The amount of time that the counter takes to expire is dependent on both the DPM\_PGCNTR.CNT setting and the SYS\_CLKIN frequency according to this equation:

$$\text{Expiration Time [ms]} = \text{Cycle Count} * 16 * \text{SYS\_CLKIN Period [ms]}$$

The text in brackets indicates that both the expiration time and SYS\_CLKIN period are expressed in milliseconds. To determine the cycle count refer to the following table which shows cycle counts for the different DPM\_PGCNTR.CNT settings.

**Table 5-3:** Cycle Counts for DPM\_PGCNTR.CNT Settings

| DPM_PGCNTR.CNT Setting            | Cycle Count   |
|-----------------------------------|---------------|
| 0x0004                            | 128 (Default) |
| 0x0080                            | 4,096         |
| 0x0100                            | 8,192         |
| 0x0200                            | 16,384        |
| 0x0400                            | 32,768        |
| 0x0800                            | 65,536        |
| 0x1000                            | 131,072       |
| 0x2000                            | 262,144       |
| 0x4000                            | 524,288       |
| 0x8000                            | 1,048,576     |
| All other values not listed above | Reserved      |

The following table shows examples of different SYS\_CLKIN frequencies and DPM\_PGCNTR.CNT settings and the resulting power good counter expiration times. Note that these are example calculations only and that the SYS\_CLKIN frequency selected must be within the range allowed by the data sheet for your processor.

**Table 5-4:** Example DPM Counter Expiration Time Calculations

| DPM_PGCNTR.CNT<br>Setting | Cycle Count (Cycles) | SYS_CLKIN<br>Frequency (MHz) | SYS_CLKIN Period<br>(ms) | Expiration Time (ms) |
|---------------------------|----------------------|------------------------------|--------------------------|----------------------|
| 0x0400                    | 32,768               | 50                           | 0.00002                  | 10.45696             |
| 0x0800                    | 65,536               | 40                           | 0.000025                 | 26.2144              |
| 0x4000                    | 524,288              | 25                           | 0.00004                  | 335.54432            |

## Using SYS\_PWRGD to Ensure That VDD\_INT is Restored

Voltage regulators commonly have a power good output which can be directly connected to the SYS\_PWRGD input of the processor. However, it is the user's responsibility to make sure that when the power good output of the regulator is asserted VDD\_INT has been restored to within the operating conditions specified in the processor's data sheet. If the regulator's power good output asserts before VDD\_INT has been restored to within the operating conditions specified in the processor's data sheet then the power good counter must be used to ensure that VDD\_INT has been restored to within specifications.

When the SYS\_PWRGD signal is used to indicate VDD\_INT is restored it is suggested that DPM\_PGCNTR.CNT be left at its default setting. The DPM then waits until the default DPM\_PGCNTR.CNT setting is expired and SYS\_PWRGD is asserted before kicking off the boot process.

## DPM Event Control

The DPM event is triggered when an enabled wake-up is asserted. DPM bus errors are generated when a misaligned access to a registers occurs or when an attempt is made to access unused DPM address space or a write protected register.

### DPM Events

The DPM event interrupt is triggered when any bit in DPM\_WAKE\_STAT is set. This indicates that an enabled wake-up was asserted. The DPM event interrupt will stay active until the user clears any bits that are set in DPM\_WAKE\_STAT.

### DPM Errors

The DPM will generate a bus error if a read or write transaction is attempted to an unused address within the DPM address range or if a misaligned access is made to an existing DPM register. In addition to the bus error DPM\_STAT.ADDRERR will be set.

If a write to a write protected DPM register is attempted a bus error will also be generated. In addition, DPM\_STAT.LWERR will be set.

## Configuring DPM DEEP SLEEP Mode

The DPM DEEP SLEEP mode gates all core and system clocks in order to save power.

### *PREREQUISITE:*

DEEP SLEEP can be entered from any state in which software can run. Reading `DPM_STAT.CURMODE` shows what the current power mode is. Clocks do not stop immediately after entry to DEEP SLEEP is requested, but no further action is needed to guarantee that the transition happens. Cores need to be idle in order for clocks are shut down.

1. Read `DPM_STAT` and `CGU_STAT`
2. If `DPM_STAT.CURMODE = FULL ON`, wait for `CGU_STAT.PLLBP = 0`, `CGU_STAT.PLOCK = 1` and `CGU_STAT.CLKSALGN = 0`.
3. If `DPM_STAT.CURMODE = ACTIVE` with PLL Disabled, wait for `CGU_STAT.PLEN = 0`
4. Enable the DPM Event interrupt to wake up the desired core in order to exit idle after DEEP SLEEP.
5. Set `DPM_WAKE_POL` wake-up source bits (`WSn`) to the desired polarity.
6. Enable the desired wake-up sources in `DPM_WAKE_EN`.
7. Set `DPM_CTL.DEEPSLEEP`
8. Clear all pending core transactions (this is done with an `SSYNC` instruction in Blackfin), DMA transactions, and interrupts.
9. Go to idle in all cores.

### *RESULT:*

The DPM is now configured to DEEP SLEEP mode and can be woken by asserting any of the enabled wakeup sources.

## Configuring DPM HIBERNATE Mode

The HIBERNATE mode allows `VDD_INT` to be shut off in order to save power.

### *PREREQUISITE:*

HIBERNATE can be entered from any DPM state in which software can be run. Reading `DPM_STAT.CURMODE` shows what the current power mode is. Make sure to save any data that needs to be preserved to the `DPM_RESTOREn` registers or to external memory before placing the processor into HIBERNATE mode. The following steps should be performed in code running from L1 memory on Blackfin processors.

1. Set `DPM_WAKE_POL` wake-up source bits (`WSn`) to the desired polarity.
2. Enable the desired wake-up sources in `DPM_WAKE_EN`.
3. Write to `DPM_HIB_DIS` if any blocks not on the `VDD_INT` domain need to be disabled.

*ADDITIONAL INFORMATION:* Note that `DPM_HIB_DIS` may not exist on your processor if there are no blocks on domains other than `VDD_INT` that are capable of being disabled during hibernate.

4. Write the power good count value to `DPM_PWRGDCNT.CNT`
5. Clear all pending core transactions (this is done with an `SSYNC` instruction in Blackfin), DMA transactions, and interrupts.
6. Set the `DPM_CTL.HIBERNATE` bit

*RESULT:*

The DPM is now configured to HIBERNATE mode.

## ADSP-BF60x Wake-Up Sources

The table below shows the HIBERNATE and DEEP SLEEP wake-up sources for the ADSP-BF60x.

The first column shows which wake-up source bit (`WSn`) is used in the `DPM_WAKE_EN`, `DPM_WAKE_POL`, and `DPM_WAKE_STAT` registers. The Assigned Source column shows which peripheral or pin source is assigned to the `WSn` bit. Peripherals in parentheses mean that the source can either be used as a GPIO wake-up or as the specific peripheral wake-up listed. The DEEP SLEEP column indicates whether or not the wake-up source can wake the processor from DEEP SLEEP. The HIBERNATE column indicates whether or not the wake-up source can wake the processor from HIBERNATE.

**Table 5-5:** ADSP-BF60x HIBERNATE and DEEP SLEEP Wake-up Sources

| DPM_WAKE_EN,<br>DPM_WAKE_POL, &<br>DPM_WAKE_STAT Bit | Assigned Source | DEEP SLEEP? | HIBERNATE? |
|------------------------------------------------------|-----------------|-------------|------------|
| WS0                                                  | PA_15           | Yes         | Yes        |
| WS1                                                  | PB_15           | Yes         | Yes        |
| WS2                                                  | PC_15           | Yes         | Yes        |
| WS3                                                  | PD_06           | Yes         | Yes        |
| WS4                                                  | PE_12           | Yes         | Yes        |

**Table 5-5:** ADSP-BF60x HIBERNATE and DEEP SLEEP Wake-up Sources (Continued)

| DPM_WAKE_EN,<br>DPM_WAKE_POL, &<br>DPM_WAKE_STAT Bit | Assigned Source                                           | DEEP SLEEP? | HIBERNATE? |
|------------------------------------------------------|-----------------------------------------------------------|-------------|------------|
| WS5                                                  | PG_04 (CAN0_RX)                                           | Yes         | Yes        |
| WS6                                                  | PG_13                                                     | Yes         | Yes        |
| WS7                                                  | USB<br>(DPM_WAKE_POL.WS<br>7 must be set to 1 for<br>USB) | No          | Yes        |
| WS30: WS8                                            | RESERVED                                                  | NA          | NA         |

## ADSP-BF60x Clock Buffer Disable Bit Assignments

The table below shows the clock buffers that may be disabled on ADSP-BF60x.

The first column shows which system clock buffer disable bit (SCBF<sub>n</sub>) is used in the DPM\_SCBF\_DIS register. The Assigned Clock column shows which clock buffer is assigned to the SCBF<sub>n</sub> bit.

**Table 5-6:** ADSP-BF60x Bit Assignments for DPM\_SCBF\_DIS

| DPM_SCBF_DIS Bit | Assigned Clock |
|------------------|----------------|
| SCBF0            | SCLK0          |
| SCBF1            | SCLK1          |
| SCBF2            | DCLK           |
| SCBF3            | OUTCLK         |

## ADSP-BF60x Hibernate Disable Bit Assignments

The table below shows the functional units that may be disabled during hibernate on ADSP-BF60x processors.

The first column shows which hibernate disable bit (HD<sub>n</sub>) is used in the DPM\_HIB\_DIS register. The functional unit column shows which functional unit is assigned to the HD<sub>n</sub> bit to allow disabling upon entering hibernate.

**Table 5-7:** ADSP-BF60x Bit Assignments for DPM\_HIB\_DIS

| DPM_SCBF_DIS Bit | Functional Unit           |
|------------------|---------------------------|
| HD0              | System crystal oscillator |
| HD7 : HD1        | Reserved                  |

## DPM Register Descriptions

Dynamic Power Management (DPM) contains the following registers.

**Table 5-8:** DPM Module Registers

| Name               | Description                              |
|--------------------|------------------------------------------|
| DPM_CTL            | Control Register                         |
| DPM_STAT           | Status Register                          |
| DPM_CCBF_DIS       | Core Clock Buffer Disable Register       |
| DPM_CCBF_EN        | Core Clock Buffer Enable Register        |
| DPM_CCBF_STAT      | Core Clock Buffer Status Register        |
| DPM_CCBF_STAT_STKY | Core Clock Buffer Status Sticky Register |
| DPM_SCBF_DIS       | System Clock Buffer Disable Register     |
| DPM_WAKE_EN        | Wakeup Enable Register                   |
| DPM_WAKE_POL       | Wakeup Polarity Register                 |
| DPM_WAKE_STAT      | Wakeup Status Register                   |
| DPM_HIB_DIS        | Hibernate Disable Register               |
| DPM_PGCNTR         | Power Good Counter Register              |
| DPM_RESTOREn       | Restore n Register                       |



## Control Register

The DPM\_CTL register controls sleep modes selections and PLL operations of the DPM. A write protect feature permits locking out changes to this register.

### DPM\_CTL: Control Register - R/W

Reset = 0x0000 0000

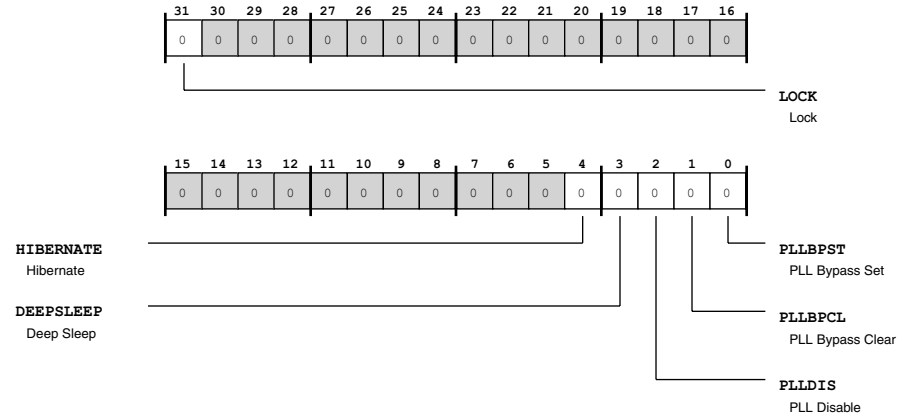


**Figure 5-2:** DPM\_CTL Register Diagram

**Table 5-9:** DPM\_CTL Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                     |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK      | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_CTL . LOCK bit is set, the DPM_CTL register is read only (locked).                                                                 |
|         |           | 0      Unlock                                                                                                                                                                                               |
|         |           | 1      Lock                                                                                                                                                                                                 |
| 4       | HIBERNATE | Hibernate.<br>The DPM_CTL . HIBERNATE bit puts the DPM into hibernate mode. The DPM stays in this mode until a wakeup event or reset occurs. For more information about DPM modes, see the operating modes. |
|         |           | 0      No Action                                                                                                                                                                                            |
|         |           | 1      Hibernate                                                                                                                                                                                            |

**Table 5-9:** DPM\_CTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                         |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | DEEPSLEEP | Deep Sleep.<br>The DPM_CTL.DEEPSLEEP bit puts the DPM into deep sleep mode. The DPM stays in this mode until a wakeup event occurs. For more information about DPM modes, see the functional description.                                       |
|         |           | 0 No Action                                                                                                                                                                                                                                     |
|         |           | 1 Deep Sleep                                                                                                                                                                                                                                    |
| 2       | PLLDIS    | PLL Disable.<br>While the DPM is in active mode, it is possible to disable the PLL with the DPM_CTL.PLLDIS bit, keeping the DPM active and running with lower power consumption. For more information about DPM modes, see the operating modes. |
|         |           | 0 Enable                                                                                                                                                                                                                                        |
|         |           | 1 Disable                                                                                                                                                                                                                                       |
| 1       | PLLBPCL   | PLL Bypass Clear.<br>While the DPM is in active mode, it is possible to disable the PLL bypass with the DPM_CTL.PLLBPCL bit, transitioning the DPM to full on mode. For more information about DPM modes, see the operating modes.              |
|         |           | 0 No action                                                                                                                                                                                                                                     |
|         |           | 1 Disable PLL Bypass                                                                                                                                                                                                                            |
| 0       | PLLB PST  | PLL Bypass Set.<br>While the DPM is in full on mode, it is possible to enable the PLL bypass with the DPM_CTL.PLLBPST bit, transitioning the DPM to active mode. For more information about DPM modes, see the operating modes.                 |
|         |           | 0 No action                                                                                                                                                                                                                                     |
|         |           | 1 Enable PLL Bypass                                                                                                                                                                                                                             |

## Status Register

The DPM\_STAT register indicates error status for DPM operations and indicates the current and previous DPM modes.

### DPM\_STAT: Status Register - R/W

Reset = 0x0000 0001

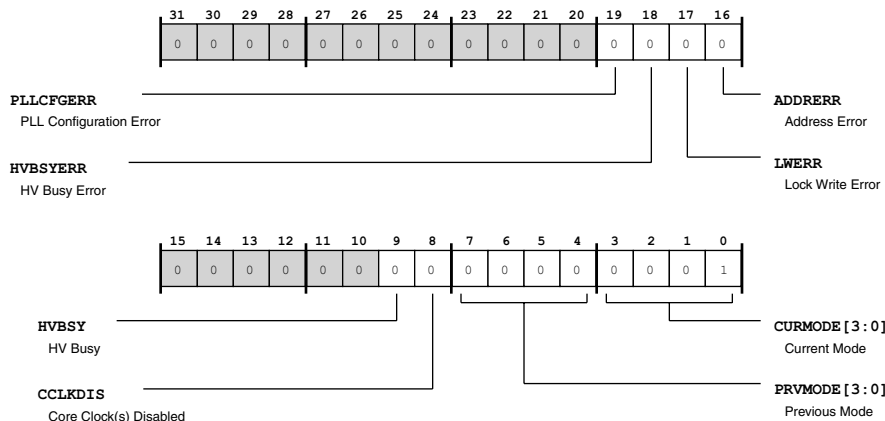


**Figure 5-3:** DPM\_STAT Register Diagram

**Table 5-10:** DPM\_STAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                            |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19      | PLLCFGERR | PLL Configuration Error.<br>The DPM_STAT.PLLCFGERR bit indicates that either the DPM_CTL.PLLBPST bit and DPM_CTL.PLLBPCL bit have been set (=1) simultaneously or the DPM_CTL.PLLBPST has been set while the DPM is in full on mode or while the DPM is entering full on mode (DPM_CTL.PLLBPCL=1). |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                        |
|         |           | 1 PLL Configuration Error                                                                                                                                                                                                                                                                          |
| 18      | HVBSYERR  | HV Busy Error.<br>The DPM_STAT.HVBSYERR bit indicates that a read access of DPM_WAKE_STAT or DPM_RESTOREn occurred during restore of DPM LV from DPM HV                                                                                                                                            |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                        |
|         |           | 1 HV Busy Error                                                                                                                                                                                                                                                                                    |

**Table 5-10:** DPM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | LWERR    | Lock Write Error.<br>The DPM_STAT.LWERR bit indicates an attempt to write to a locked DPM register while the global lock bit is set (SPU_CTL_GLCK bit =1).                                                                                                                                           |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                     |
|         |          | 1                      Lock Write Error                                                                                                                                                                                                                                                              |
| 16      | ADDRERR  | Address Error.<br>The DPM_STAT.ADDRERR bit indicates either an attempted read or write access to an address in the DPM MMR space that are not implemented addresses, an attempted write access to a DPM read only register, or a read or write access to a non aligned address in the DPM MMR space. |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                     |
|         |          | 1                      Address Error                                                                                                                                                                                                                                                                 |
| 9       | HVBSY    | HV Busy.<br>The DPM_STAT.HVBSY bit indicates whether the DPM HV unit is ready (has completed all transactions) or is busy (transfers from HV have not yet completed).                                                                                                                                |
|         |          | 0                      Ready                                                                                                                                                                                                                                                                         |
|         |          | 1                      Busy                                                                                                                                                                                                                                                                          |
| 8       | CCLKDIS  | Core Clock(s) Disabled.<br>The DPM_STAT.CCLKDIS bit indicates whether or not one or more of the core clocks is disabled.                                                                                                                                                                             |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                     |
|         |          | 1                      Clock (1 or more) Disabled                                                                                                                                                                                                                                                    |

**Table 5-10:** DPM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | PRVMODE  | Previous Mode.<br>The <code>DPM_STAT.PRVMODE</code> bits indicate the previous DPM mode of operation. All values not shown are reserved. |
|         |          | 0                      Reset                                                                                                             |
|         |          | 1                      Full-On                                                                                                           |
|         |          | 2                      Active                                                                                                            |
|         |          | 3                      Active with PLL disabled                                                                                          |
|         |          | 4                      Deep Sleep                                                                                                        |
|         |          | 5                      Hibernate                                                                                                         |
| 3:0     | CURMODE  | Current Mode.<br>The <code>DPM_STAT.CURMODE</code> bits indicate the current DPM mode of operation. All values not shown are reserved.   |
|         |          | 1                      Full-On                                                                                                           |
|         |          | 2                      Active                                                                                                            |
|         |          | 3                      Active with PLL disabled                                                                                          |

## Core Clock Buffer Disable Register

The `DPM_CCBF_DIS` register controls the core n clock buffers. The number of clocks varies with the processor design, with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. This register includes a write protection lock.

DPM\_CCBF\_DIS: Core Clock Buffer Disable Register - R/W

Reset = 0x0000 0000

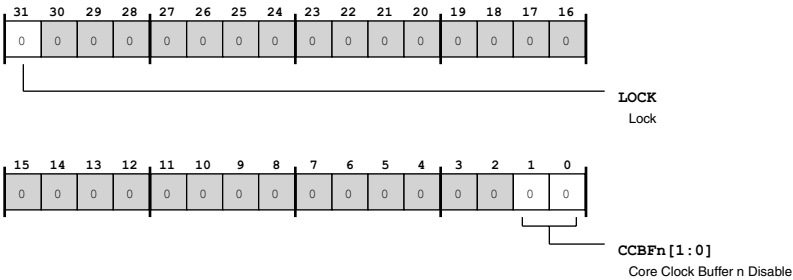


Figure 5-4: DPM\_CCBF\_DIS Register Diagram

Table 5-11: DPM\_CCBF\_DIS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                    |                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 31      | LOCK     | Lock.                                                                                                                                                                                                      |                |
|         |          | 0                                                                                                                                                                                                          | Unlock         |
|         |          | 1                                                                                                                                                                                                          | Lock           |
| 1:0     | CCBFn    | Core Clock Buffer n Disable.<br>The DPM_CCBF_DIS.CCBFn bits provide a core clock buffer disable for each core on the processor with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. |                |
|         |          | 0                                                                                                                                                                                                          | No Action      |
|         |          | 1                                                                                                                                                                                                          | Disable Buffer |

Core Clock Buffer Enable Register

The DPM\_CCBF\_EN register controls the core n clock buffers. The number of clocks varies with the processor design, with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. This register includes a write protection lock.

**DPM\_CCBF\_EN: Core Clock Buffer Enable Register - R/W**

Reset = 0x0000 0000

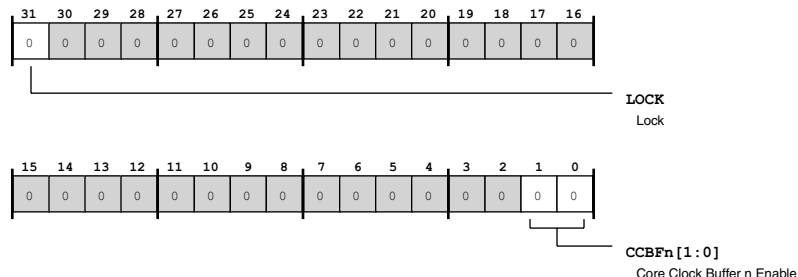


**Figure 5-5:** DPM\_CCBF\_EN Register Diagram

**Table 5-12:** DPM\_CCBF\_EN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_CCBF_EN . LOCK bit is set, the DPM_CCBF_EN register is read only (locked).                                                      |
|         |          | 0      Unlock                                                                                                                                                                                            |
|         |          | 1      Lock                                                                                                                                                                                              |
| 1:0     | CCBFn    | Core Clock Buffer n Enable.<br>The DPM_CCBF_EN .CCBFn bits provide a core clock buffer enable for each core on the processor with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. |
|         |          | 0      No Action                                                                                                                                                                                         |
|         |          | 1      Enable Buffer                                                                                                                                                                                     |

## Core Clock Buffer Status Register

The DPM\_CCBF\_STAT register indicates core clock buffer enable or disable status for each core on the processor, with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on.

DPM\_CCBF\_STAT: Core Clock Buffer Status Register - R/NW

Reset = 0x0000 0000

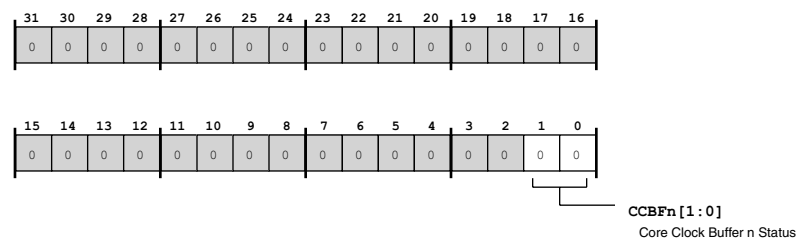


Figure 5-6: DPM\_CCBF\_STAT Register Diagram

Table 5-13: DPM\_CCBF\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                       |                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 1:0     | CCBFn    | Core Clock Buffer n Status.<br>The DPM_CCBF_STAT.CCBFn bits indicates core clock buffer enabled or disabled status for each core on the processor with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. |                 |
|         |          | 0                                                                                                                                                                                                                             | Buffer Enabled  |
|         |          | 1                                                                                                                                                                                                                             | Buffer Disabled |

Core Clock Buffer Status Sticky Register

The DPM\_CCBF\_STAT\_STKY register indicates core n clock buffer enable or disable sticky status for each core on the processor, with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on.

DPM\_CCBF\_STAT\_STKY: Core Clock Buffer Status Sticky Register - R/W

Reset = 0x0000 0000

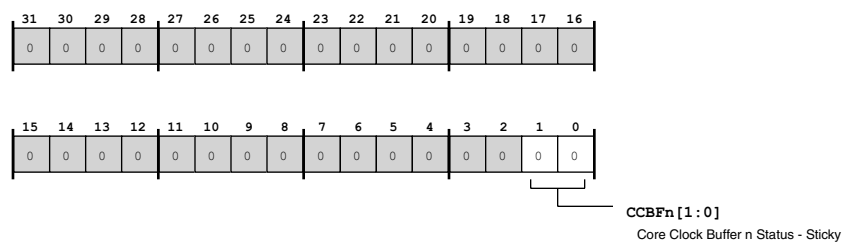


Figure 5-7: DPM\_CCBF\_STAT\_STKY Register Diagram



**Table 5-14:** DPM\_CCBF\_STAT\_STKY Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1:0     | CCBFn    | Core Clock Buffer n Status - Sticky.<br>The DPM_CCBF_STAT_STKY.CCBFn bits indicates core clock buffer enabled or disabled sticky status for each core on the processor with bit 0 corresponding to CCLK0, bit 1 corresponding to CCLK1, and so on. The sticky status shows that the status was set since the last time the bit was cleared with a W1A or reset. |
|         |          | 0 Buffer Enabled - Sticky                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Buffer Disabled - Sticky                                                                                                                                                                                                                                                                                                                                      |

## System Clock Buffer Disable Register

The DPM\_SCBF\_DIS register controls the system n clock buffers. The number of clocks varies with the processor design. See the DPM Clock Buffer Disable Bit Assignments topic for the bit assignments of this processor.

**DPM\_SCBF\_DIS: System Clock Buffer Disable Register - R/W**

Reset = 0x0000 0000

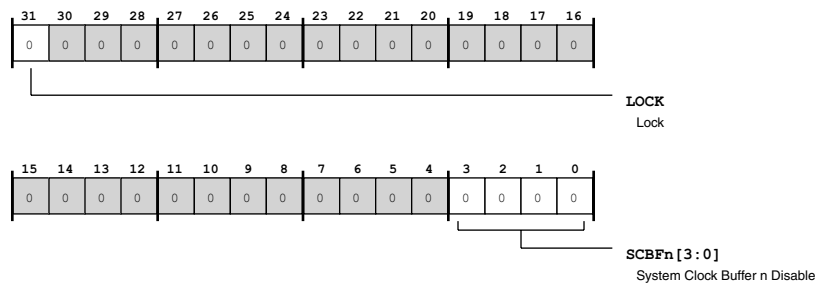


**Figure 5-8:** DPM\_SCBF\_DIS Register Diagram

Table 5-15: DPM\_SCBF\_DIS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_SCBF_DIS . LOCK bit is set, the DPM_SCBF_DIS register is read only (locked).                                                                                                |
|         |          | 0                      Unlock                                                                                                                                                                                                                        |
|         |          | 1                      Lock                                                                                                                                                                                                                          |
| 3:0     | SCBFn    | System Clock Buffer n Disable.<br>The DPM_SCBF_DIS . SCBFn bits provide a system clock buffer enable for each system clock domain on the processor. See theDPM Clock Buffer Disable Bit Assignments topic for the bit assignments of this processor. |
|         |          | 0                      Enable Buffer                                                                                                                                                                                                                 |
|         |          | 1                      Disable Buffer                                                                                                                                                                                                                |

Wakeup Enable Register

The DPM\_WAKE\_EN register enables the wakeup event sources for exiting deep sleep mode. The number of wakeup sources varies with the processor design, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. This register includes a write protection lock. For information about wakeup source assignments, see the DPM Wakeup Sources topic.

DPM\_WAKE\_EN: Wakeup Enable Register - R/W

Reset = 0x0000 0000

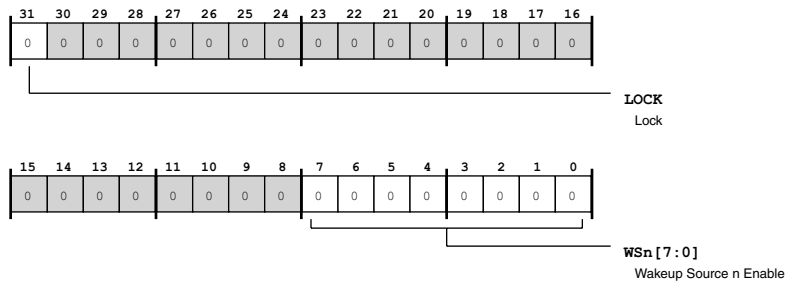


Figure 5-9: DPM\_WAKE\_EN Register Diagram

**Table 5-16:** DPM\_WAKE\_EN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_WAKE_EN . LOCK bit is set, the DPM_WAKE_EN register is read only (locked).                                                                                                                                         |
|         |          | 0                      Unlock                                                                                                                                                                                                                                                               |
|         |          | 1                      Lock                                                                                                                                                                                                                                                                 |
| 7:0     | WSn      | Wakeup Source n Enable.<br>The DPM_WAKE_EN . WSn bits enable wakeup sources for exiting deep sleep mode, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. For information about wakeup source assignments, see the DPM Wakeup Sources topic. |
|         |          | 0                      Disable Wakeup Source                                                                                                                                                                                                                                                |
|         |          | 1                      Enable Wakeup Source                                                                                                                                                                                                                                                 |

## Wakeup Polarity Register

The DPM\_WAKE\_POL register select polarity (active high or low) of the wakeup event sources for exiting deep sleep mode. The number of wakeup sources varies with the processor design, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. This register includes a write protection lock. For information about wakeup source assignments, see the DPM Wakeup Sources topic.

**DPM\_WAKE\_POL: Wakeup Polarity Register - R/W**

Reset = 0x0000 0000

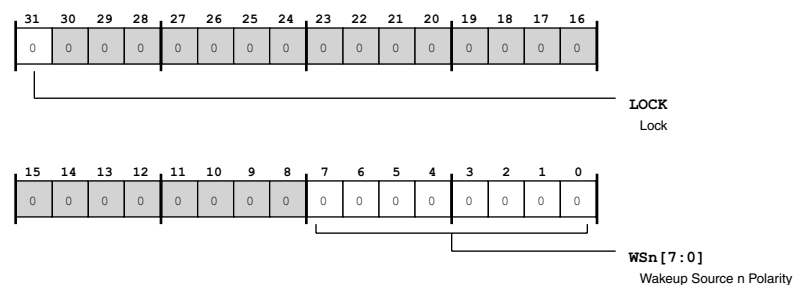


**Figure 5-10:** DPM\_WAKE\_POL Register Diagram

Table 5-17: DPM\_WAKE\_POL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_WAKE_POL . LOCK bit is set, the DPM_WAKE_POL register is read only (locked).                                                                                                                                                                           |
|         |          | 0                      Unlock                                                                                                                                                                                                                                                                                                   |
|         |          | 1                      Lock                                                                                                                                                                                                                                                                                                     |
| 7:0     | WSn      | Wakeup Source n Polarity.<br>The DPM_WAKE_POL . WSn bits select polarity (active high or low) of wakeup sources for exiting deep sleep mode, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. For information about wakeup source assignments, see the DPM Wakeup Sources topic. |
|         |          | 0                      Low Active Wakeup                                                                                                                                                                                                                                                                                        |
|         |          | 1                      High Active Wakeup                                                                                                                                                                                                                                                                                       |

Wakeup Status Register

The DPM\_WAKE\_STAT register indicates the enabled and active status of the wakeup event sources for exiting deep sleep mode. The number of wakeup sources varies with the processor design, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. For information about wakeup source assignments, see the DPM Wakeup Sources topic.

DPM\_WAKE\_STAT: Wakeup Status Register - R/W

Reset = 0x0000 0000

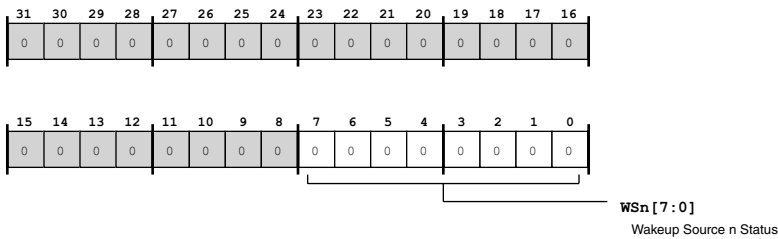


Figure 5-11: DPM\_WAKE\_STAT Register Diagram

**Table 5-18:** DPM\_WAKE\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | WSn      | Wakeup Source n Status.<br>The DPM_WAKE_STAT.WSn bits indicate the enabled and active status of wakeup sources for exiting deep sleep mode, with bit 0 corresponding to wakeup source 0, bit 1 corresponding to wakeup source 1, and so on. For information about wakeup source assignments, see the DPM Wakeup Sources topic. |
|         |          | 0      No Status                                                                                                                                                                                                                                                                                                               |
|         |          | 1      Enabled and Active                                                                                                                                                                                                                                                                                                      |

Hibernate Disable Register

The DPM\_HIB\_DIS register controls which functional units on the V<sub>DD\_EXT</sub> domain are disabled during hibernate. For the bit assignments of this processor, see the DPM Hibernate Disable Bit Assignments topic.

**DPM\_HIB\_DIS: Hibernate Disable Register - R/W**

Reset = 0x0000 0000

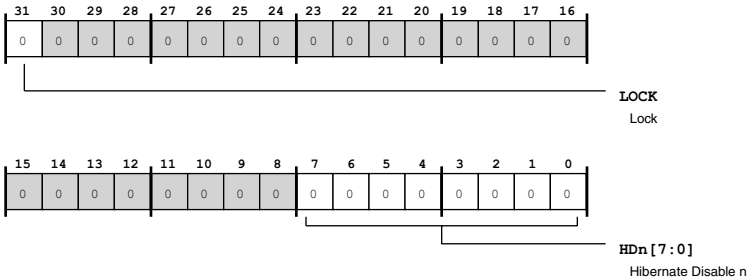


**Figure 5-12:** DPM\_HIB\_DIS Register Diagram

**Table 5-19:** DPM\_HIB\_DIS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_HIB_DIS.LOCK bit is set, the DPM_HIB_DIS register is read only (locked).                                                                                                |
|         |          | 0                      Unlock                                                                                                                                                                                                                    |
|         |          | 1                      Lock                                                                                                                                                                                                                      |
| 7:0     | HDn      | Hibernate Disable n.<br>The DPM_HIB_DIS.HDn bits control which functional units on the V <sub>DD_EXT</sub> domain are disabled during hibernate. For the bit assignments of this processor, see the DPM Hibernate Disable Bit Assignments topic. |
|         |          | 0                      Enable External Wakeup Source                                                                                                                                                                                             |
|         |          | 1                      Disable Functional Unit During Hibernate                                                                                                                                                                                  |

## Power Good Counter Register

The DPM\_PGCNTR register selects the count of CLKIN cycles the DPM waits for the V<sub>DDINT</sub> power supply to reach the specified value. This register includes a write protection lock.

**DPM\_PGCNTR: Power Good Counter Register - R/W**

Reset = 0x0000 0004

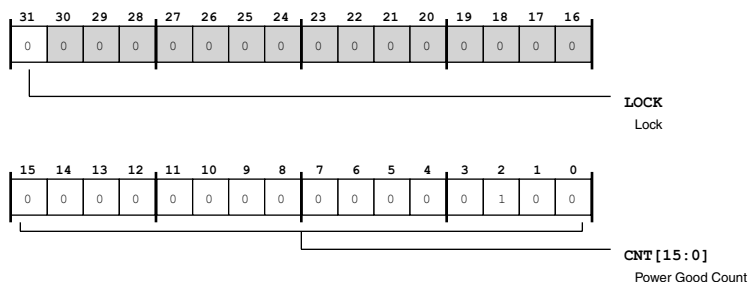


**Figure 5-13:** DPM\_PGCNTR Register Diagram

**Table 5-20:** DPM\_PGCNTR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the DPM_PGCNTR.LOCK bit is set, the DPM_PGCNTR register is read only (locked).                                                           |
|         |          | 0                      Unlock                                                                                                                                                                             |
|         |          | 1                      Lock                                                                                                                                                                               |
| 15:0    | CNT      | Power Good Count.<br>The DPM_PGCNTR.CNT bits select the count that the DPM uses to determine that the $V_{DDINT}$ supply has reached the specified value. All values other than those shown are reserved. |
|         |          | 4                      128 cycles                                                                                                                                                                         |
|         |          | 128                    4K cycles                                                                                                                                                                          |
|         |          | 256                    8K cycles                                                                                                                                                                          |
|         |          | 512                    16K cycles                                                                                                                                                                         |
|         |          | 1024                   32K cycles                                                                                                                                                                         |
|         |          | 2048                   64K cycles                                                                                                                                                                         |
|         |          | 4096                   128K cycles                                                                                                                                                                        |
|         |          | 8192                   256K cycles                                                                                                                                                                        |
|         |          | 16384                   512K cycles                                                                                                                                                                       |
|         |          | 32768                   1M cycles                                                                                                                                                                         |

## Restore n Register

The DPM\_RESTOREn registers are general-purpose registers that the DPM preserves during hibernate mode and restores on exit from hibernate. The usage of these registers is application specific and does not affect DPM operations. For more information about using the DPM\_RESTOREn registers, see the programming model.

DPM\_RESTOREn: Restore n Register - R/W

Reset = 0x0000 0000

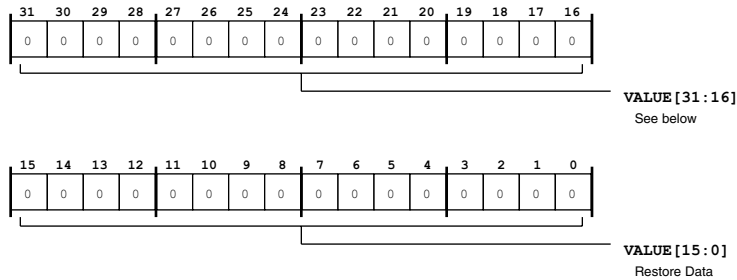


Figure 5-14: DPM\_RESTOREn Register Diagram

Table 5-21: DPM\_RESTOREn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Restore Data.<br>The DPM_RESTOREn.VALUE data is stored during hibernate mode and restored by the DPM on exit from hibernate. |



## 6 Core Timers

Each processor core has its own dedicated timer. The core timer is clocked by the internal processor clock and is typically used as a system tick clock for generating periodic operating system interrupts.

### NOTE:

The core timer stops counting when there is an emulation event. The emulation event is controlled by the SDU (System Debug Unit). Please refer to the SDU chapter for more details on this.

## Core Timer Features

The core timer is a programmable 32-bit interval timer which can generate periodic interrupts. Unlike other peripherals, the core timer resides inside the Blackfin core and runs at the core clock (CCLK) rate. Core timer features include:

- 32-bit timer with 8-bit prescaler
- Operates at core clock (CCLK) rate
- Dedicated high-priority interrupt channel
- Single-shot or continuous operation

## TMR Register List

The core timer (TMR) is a programmable 32-bit interval timer, which can generate periodic interrupts. Unlike other peripherals, the core timer resides inside the processor core and runs at the core clock (CCLK) rate. A set of registers govern TMR operations. For more information on TMR functionality, see the TMR register descriptions.

**Table 6-1:** TMR Module Registers

| Name    | Description            |
|---------|------------------------|
| TCNTL   | Timer Control Register |
| TPERIOD | Timer Period Register  |
| TSCALE  | Timer Scale Register   |

**Table 6-1:** TMR Module Registers (Continued)

| Name   | Description          |
|--------|----------------------|
| TCOUNT | Timer Count Register |

Block Diagram

The following figure shows the core timer block diagram.

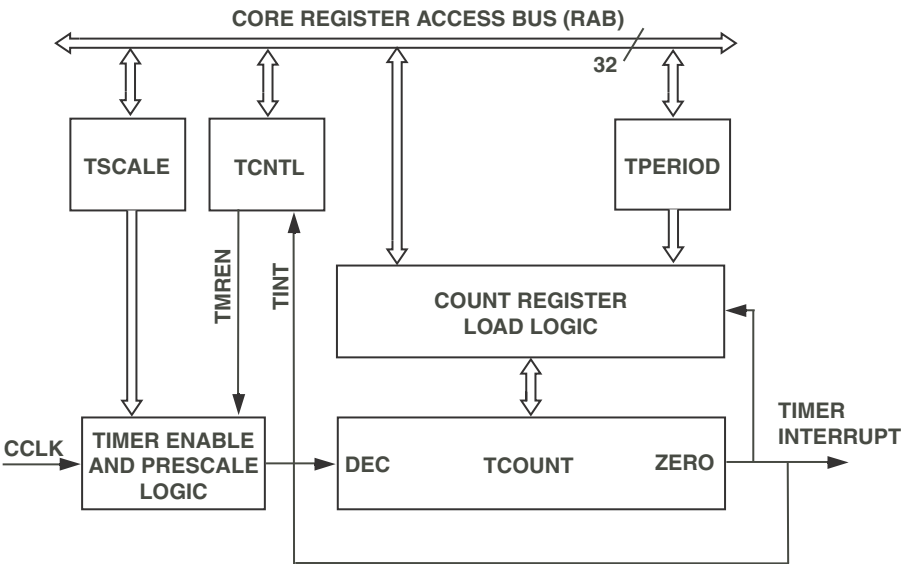


**Figure 6-1:** Core Timer Block Diagram

External Interfaces

The core timer does not directly interact with any pins of the chip.

Internal Interfaces

The core timer is accessed through the 32-bit register access bus (RAB). The module is clocked by the core clock CCLK. The timer’s dedicated interrupt request is a higher priority than requests from all other peripherals.

Timer Operation

The software should initialize the TCOUNT register before the timer is enabled. The TCOUNT register can be written directly, but writes to the TPERIOD register are also passed through to TCOUNT.

When the timer is enabled by setting the TMREN bit in the core timer control register (TCNTL), the TCOUNT register is decremented once every time the prescaler TSCALE expires, that is, every TSCALE + 1 number of CCLK clock cycles. When the value of the TCOUNT register reaches 0, an interrupt is generated and the TINT bit is set in the TCNTL register.

If the TAUTORLD bit in the TCNTL register is set, then the TCOUNT register is reloaded with the contents of the TPERIOD register and the count begins again. If the TAUTORLD bit is not set, the timer stops operation.

The core timer can be put into low power mode by clearing the TMPWR bit in the TCNTL register. Before using the timer, set the TMPWR bit. This restores clocks to the timer unit. When TMPWR is set, the core timer may then be enabled by setting the TMREN bit in the TCNTL register.

**NOTE:** Hardware behavior is undefined if TMREN is set when TMPWR = 0.

## Interrupt Processing

The timer's dedicated interrupt request is a higher priority than requests from all other peripherals. The request goes directly to the core event controller (CEC) and does not pass through the system event controller (SEC). Therefore, the interrupt processing is also completely in the CCLK domain.

**NOTE:** The core timer interrupt request is edge-sensitive and cleared by hardware automatically as soon as the interrupt is serviced.

The TINT bit in the TCNTL register indicates that an interrupt has been generated. Note that this is not a WIC bit. Write a 0 to clear it. However, the write is optional. It is not required to clear interrupt requests. The core time module doesn't provide any further interrupt enable bit. When the timer is enabled, interrupts can be masked in the CEC controller.

## TMR Register Descriptions

Core Timer (TMR) contains the following registers.

**Table 6-2:** TMR Module Registers

| Name    | Description            |
|---------|------------------------|
| TCNTL   | Timer Control Register |
| TPERIOD | Timer Period Register  |
| TSCALE  | Timer Scale Register   |
| TCOUNT  | Timer Count Register   |

# Timer Control Register

The TCNTL register functions as the TMR control and status register.

TCNTL: Timer Control Register - R/W

Reset = 0x0000 0000

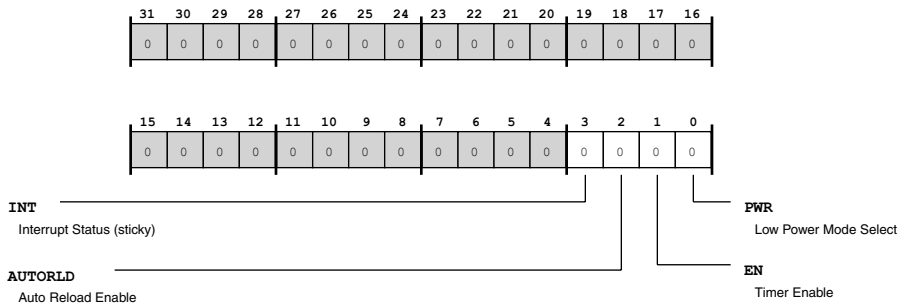


Figure 6-2: TCNTL Register Diagram

Table 6-3: TCNTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | INT      | Interrupt Status (sticky).<br>The TCNTL . INT bit indicates the status of the timer generated interrupt, which is generated when the TCOUNT value decrements to 0. This bit is sticky, remaining set until cleared.         |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                      |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                         |
| 2       | AUTORLD  | Auto Reload Enable.<br>The TCNTL . AUTORLD bit enables the TMR to automatically reload the TCOUNT register from the TPERIOD register when the count expires. If disabled, the timer stops operation when the count expires. |
|         |          | 0 Disable Auto Reload                                                                                                                                                                                                       |
|         |          | 1 Enable Auto Reload                                                                                                                                                                                                        |

**Table 6-3:** TCNTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | EN       | Timer Enable.<br>The TCNTL.EN bit enables TMR operation, starting the count at the value in the TCOUNT register and decrementing this value each time the prescaler (TSCALE) time expires.                                                                         |
|         |          | 0      Disable Timer                                                                                                                                                                                                                                               |
|         |          | 1      Enable Timer                                                                                                                                                                                                                                                |
| 0       | PWR      | Low Power Mode Select.<br>The TCNTL.PWR bit selects active mode or low power mode for TMR operation. When in low power mode, clocks to the TMR are disabled. Before enabling the timer (TCNTL.EN = 1), put the timer in active mode (restoring clocks to the TMR). |
|         |          | 0      Low Power Mode                                                                                                                                                                                                                                              |
|         |          | 1      Active Mode                                                                                                                                                                                                                                                 |

## Timer Period Register

The TPERIOD register holds the timer period. When the processor writes the TPERIOD register, the TMR automatically writes the value to the TCOUNT register also. When count auto reload is enabled (TCNTL.AUTORLD = 1), the TMR automatically reloads the TCOUNT register with the value in TPERIOD when the counter expires. Note that writes to TPERIOD are ignored when the timer is running.

**TPERIOD: Timer Period Register - R/W**

Reset = 0x0000 0000

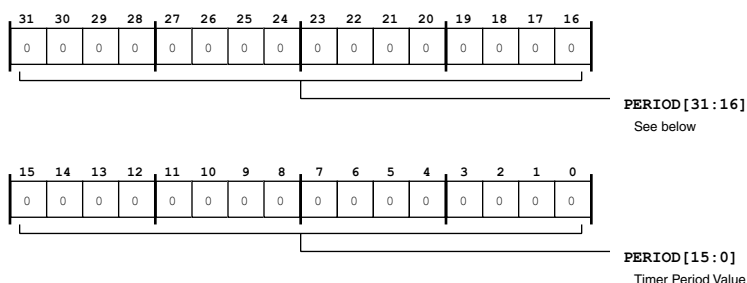


**Figure 6-3:** TPERIOD Register Diagram

Table 6-4: TPERIOD Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                     |
|---------|----------|-----------------------------------------------------------------------------|
| 31:0    | PERIOD   | Timer Period Value.<br>The TPERIOD.PERIOD bits hold the timer period value. |

Timer Scale Register

The TSCALE register stores the scaling value that is one core clock (CCLK) cycle less than the number of cycles between each decrement of the timer count (TCOUNT). For example, if the value in the TSCALE register is 0, the TCOUNT register decrements once for every CCLK cycle. If the TSCALE value is 1, the TCOUNT decrements once every two CCLK cycles.

TSCALE: Timer Scale Register - R/W  
Reset = 0x0000 0000

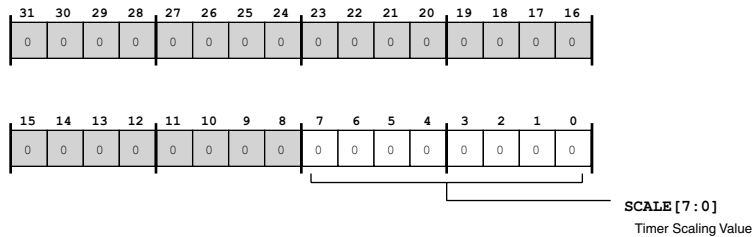


Figure 6-4: TSCALE Register Diagram

Table 6-5: TSCALE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | SCALE    | Timer Scaling Value.<br>The TSCALE.SCALE bits hold a scaling factor, relating CCLK cycles to the rate that the count decrements. |

Timer Count Register

The TCOUNT register holds the current count for the TMR. Typically, this count is initially load with a write to the TPERIOD register, because the TMR automatically copies the value written to the TPERIOD register and writes the value to the TCOUNT register. The TCOUNT register can be written directly. When the timer is running, the TCOUNT value decrements once every TSCALE + 1 core clock (CCLK) cycles. When the value in TCOUNT reaches 0, the TMR generates an interrupt and sets the TCNTL.INT bit. Note that writes to TCOUNT are ignored when the timer is running.

TCOUNT: Timer Count Register - R/W

Reset = 0x0000 0000

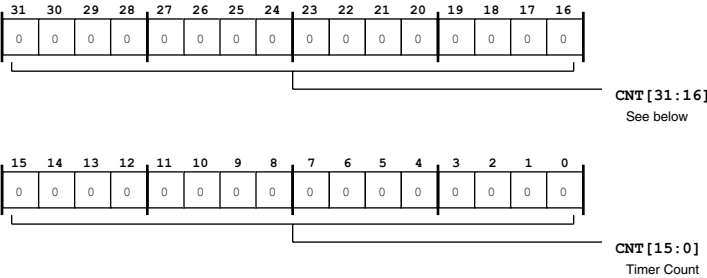


Figure 6-5: TCOUNT Register Diagram

Table 6-6: TCOUNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                 |
|---------|----------|-------------------------------------------------------------------------|
| 31:0    | CNT      | Timer Count.<br>The TCOUNT.CNT bits hold the current timer count value. |





# 7 System Event Controller (SEC)

System event management is the responsibility of the system event controller (SEC). The SEC manages the configuration of all system event sources as well as the propagation of system events to all connected cores and the system fault interface.

## SEC Features

The following list describes the system event controller features.

- Comprehensive system event source management including interrupt enable, fault enable, prioritization, core mapping and source grouping.
- Fault management including fault action configuration, time out, external indication, and system reset.
- Determinism where all system events have the same propagation delay and provide unique identification of a specific system event source.
- Distributed programming model where each system event source control and all status fields are completely independent of all others.
- Slave Control Port which provides access to all SEC registers for configuration, status, and interrupt/fault service model.
- Global locking supports a register level protection model to prevent writes to “locked” registers.

## SEC Functional Description

The following sections provide a functional description of the SEC.

### SEC Register List

The system event controller (SEC) manages the system interrupt and system fault sources. The SEC also provides all system interrupt and fault sources control features, such as enable/disable, prioritization, and active/pending source status. On multi-core processors, the SEC provides connected core(s) and fault management with source pending and active indication. For more information on SEC functionality, see the SEC register descriptions.

**Table 7-1:** SEC Module Registers

| Name           | Description                               |
|----------------|-------------------------------------------|
| SEC_CCTLn      | SCI Control Register n                    |
| SEC_CSTATn     | SCI Status Register n                     |
| SEC_CPNDn      | Core Pending Register n                   |
| SEC_CACTn      | SCI Active Register n                     |
| SEC_CPMSKn     | SCI Priority Mask Register n              |
| SEC_CGMSKn     | SCI Group Mask Register n                 |
| SEC_CPLVLn     | SCI Priority Level Register n             |
| SEC_CSIDn      | SCI Source ID Register n                  |
| SEC_FCTL       | Fault Control Register                    |
| SEC_FSTAT      | Fault Status Register                     |
| SEC_FSID       | Fault Source ID Register                  |
| SEC_FEND       | Fault End Register                        |
| SEC_FDLY       | Fault Delay Register                      |
| SEC_FDLY_CUR   | Fault Delay Current Register              |
| SEC_FSRDLY     | Fault System Reset Delay Register         |
| SEC_FSRDLY_CUR | Fault System Reset Delay Current Register |
| SEC_FCOPP      | Fault COP Period Register                 |
| SEC_FCOPP_CUR  | Fault COP Period Current Register         |
| SEC_GCTL       | Global Control Register                   |
| SEC_GSTAT      | Global Status Register                    |
| SEC_RAISE      | Global Raise Register                     |
| SEC_END        | Global End Register                       |

**Table 7-1:** SEC Module Registers (Continued)

| Name       | Description               |
|------------|---------------------------|
| SEC_SCTLn  | Source Control Register n |
| SEC_SSTATn | Source Status Register n  |

## ADSP-BF60x Interrupt List

**Table 7-2:** ADSP-BF60x Interrupt List

| Description                                    | ID | DMA Channel | Sensitivity |
|------------------------------------------------|----|-------------|-------------|
| SEC0 Error                                     | 0  |             | LEVEL       |
| CGU0 Event                                     | 1  |             | PULSE/EDGE  |
| WDOG0 Expiration                               | 2  |             | LEVEL       |
| WDOG1 Expiration                               | 3  |             | LEVEL       |
| L2CTL0 ECC Error                               | 4  |             | LEVEL       |
| Core 0 Double Fault                            | 6  |             | PULSE/EDGE  |
| Core 1 Double Fault                            | 7  |             | PULSE/EDGE  |
| Core 0 Hardware Error                          | 8  |             | PULSE/EDGE  |
| Core 1 Hardware Error                          | 9  |             | PULSE/EDGE  |
| Core 0 Unhandled NMI or L1 Memory Parity Error | 10 |             | PULSE/EDGE  |
| Core 1 Unhandled NMI or L1 Memory Parity Error | 11 |             | PULSE/EDGE  |
| TIMER0 Timer 0                                 | 12 |             | LEVEL       |
| TIMER0 Timer 1                                 | 13 |             | LEVEL       |
| TIMER0 Timer 2                                 | 14 |             | LEVEL       |
| TIMER0 Timer 3                                 | 15 |             | LEVEL       |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description                    | ID | DMA Channel | Sensitivity |
|--------------------------------|----|-------------|-------------|
| TIMER0 Timer 4                 | 16 |             | LEVEL       |
| TIMER0 Timer 5                 | 17 |             | LEVEL       |
| TIMER0 Timer 6                 | 18 |             | LEVEL       |
| TIMER0 Timer 7                 | 19 |             | LEVEL       |
| TIMER0 Status                  | 20 |             | LEVEL       |
| PINT0 Pin Interrupt Block      | 21 |             | LEVEL       |
| PINT1 Pin Interrupt Block      | 22 |             | LEVEL       |
| PINT2 Pin Interrupt Block      | 23 |             | LEVEL       |
| PINT3 Pin Interrupt Block      | 24 |             | LEVEL       |
| PINT4 Pin Interrupt Block      | 25 |             | LEVEL       |
| PINT5 Pin Interrupt Block      | 26 |             | LEVEL       |
| CNT0 Status                    | 27 |             | LEVEL       |
| PWM0 PWMTMR Group              | 28 |             | LEVEL       |
| PWM0 Trip                      | 29 |             | LEVEL       |
| PWM1 PWMTMR Group              | 30 |             | LEVEL       |
| PWM1 Trip                      | 31 |             | LEVEL       |
| TWI0 Data Interrupt            | 32 |             | LEVEL       |
| TWI1 Data Interrupt            | 33 |             | LEVEL       |
| Software-driven Interrupt<br>0 | 34 |             | PULSE/EDGE  |
| Software-driven Interrupt<br>1 | 35 |             | PULSE/EDGE  |
| Software-driven Interrupt<br>2 | 36 |             | PULSE/EDGE  |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description                 | ID | DMA Channel | Sensitivity |
|-----------------------------|----|-------------|-------------|
| Software-driven Interrupt 3 | 37 |             | PULSE/EDGE  |
| ACM0 Event Miss             | 38 |             | LEVEL       |
| ACM0 Event Complete         | 39 |             | LEVEL       |
| CAN0 Receive                | 40 |             | LEVEL       |
| CAN0 Transmit               | 41 |             | LEVEL       |
| CAN0 Status                 | 42 |             | LEVEL       |
| SPORT0 Channel A DMA        | 43 | 0           | LEVEL       |
| SPORT0 Channel A Status     | 44 |             | LEVEL       |
| SPORT0 Channel B DMA        | 45 | 1           | LEVEL       |
| SPORT0 Channel B Status     | 46 |             | LEVEL       |
| SPORT1 Channel A DMA        | 47 | 2           | LEVEL       |
| SPORT1 Channel A Status     | 48 |             | LEVEL       |
| SPORT1 Channel B DMA        | 49 | 3           | LEVEL       |
| SPORT1 Channel B Status     | 50 |             | LEVEL       |
| SPORT2 Channel A DMA        | 51 | 4           | LEVEL       |
| SPORT2 Channel A Status     | 52 |             | LEVEL       |
| SPORT2 Channel B DMA        | 53 | 5           | LEVEL       |
| SPORT2 Channel B Status     | 54 |             | LEVEL       |
| SPI0 TX DMA Channel         | 55 | 6           | LEVEL       |
| SPI0 RX DMA Channel         | 56 | 7           | LEVEL       |
| SPI0 Status                 | 57 |             | LEVEL       |
| SPI1 TX DMA Channel         | 58 | 8           | LEVEL       |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| SPI1 RX DMA Channel | 59 | 9           | LEVEL       |
| SPI1 Status         | 60 |             | LEVEL       |
| RSI0 DMA Channel    | 61 | 10          | LEVEL       |
| RSI0 Interrupt 0    | 62 |             | LEVEL       |
| RSI0 Interrupt 1    | 63 |             | LEVEL       |
| SDU0 DMA            | 64 | 11          | LEVEL       |
| Reserved            | 65 | 12          |             |
| Reserved            | 66 |             |             |
| Reserved            | 67 |             |             |
| EMAC0 Status        | 68 |             | LEVEL       |
| Reserved            | 69 |             |             |
| EMAC1 Status        | 70 |             | LEVEL       |
| Reserved            | 71 |             |             |
| LP0 DMA Channel     | 72 | 13          | LEVEL       |
| LP0 Status          | 73 |             | LEVEL       |
| LP1 DMA Channel     | 74 | 14          | LEVEL       |
| LP1 Status          | 75 |             | LEVEL       |
| LP2 DMA Channel     | 76 | 15          | LEVEL       |
| LP2 Status          | 77 |             | LEVEL       |
| LP3 DMA Channel     | 78 | 16          | LEVEL       |
| LP3 Status          | 79 |             | LEVEL       |
| UART0 Transmit DMA  | 80 | 17          | LEVEL       |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description                                                 | ID | DMA Channel | Sensitivity |
|-------------------------------------------------------------|----|-------------|-------------|
| UART0 Receive DMA                                           | 81 | 18          | LEVEL       |
| UART0 Status                                                | 82 |             | LEVEL       |
| UART1 Transmit DMA                                          | 83 | 19          | LEVEL       |
| UART1 Receive DMA                                           | 84 | 20          | LEVEL       |
| UART1 Status                                                | 85 |             | LEVEL       |
| Memory DMA Stream 0<br>Source / CRC0 Input<br>Channel       | 86 | 21          | LEVEL       |
| Memory DMA Stream 0<br>Destination / CRC0 Output<br>Channel | 87 | 22          | LEVEL       |
| CRC0 Datacount<br>expiration                                | 88 |             | LEVEL       |
| CRC0 Error                                                  | 89 |             | LEVEL       |
| Memory DMA Stream 1<br>Source / CRC1 Input<br>Channel       | 90 | 23          | LEVEL       |
| Memory DMA Stream 1<br>Destination / CRC1 Output<br>Channel | 91 | 24          | LEVEL       |
| CRC1 Datacount<br>expiration                                | 92 |             | LEVEL       |
| CRC1 Error                                                  | 93 |             | LEVEL       |
| Memory DMA Stream 2<br>Source Channel                       | 94 | 25          | LEVEL       |
| Memory DMA Stream 2<br>Destination Channel                  | 95 | 26          | LEVEL       |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description                             | ID  | DMA Channel | Sensitivity |
|-----------------------------------------|-----|-------------|-------------|
| Memory DMA Stream 3 Source Channel      | 96  | 27          | LEVEL       |
| Memory DMA Stream 3 Destination Channel | 97  | 28          | LEVEL       |
| EPPI0 Channel 0 DMA                     | 98  | 29          | LEVEL       |
| EPPI0 Channel 1 DMA                     | 99  | 30          | LEVEL       |
| EPPI0 Status                            | 100 |             | LEVEL       |
| EPPI2 Channel 0 DMA                     | 101 | 31          | LEVEL       |
| EPPI2 Channel 1 DMA                     | 102 | 32          | LEVEL       |
| EPPI2 Status                            | 103 |             | LEVEL       |
| EPPI1 Channel 0 DMA                     | 104 | 33          | LEVEL       |
| EPPI1 Channel 1 DMA                     | 105 | 34          | LEVEL       |
| EPPI1 Status                            | 106 |             | LEVEL       |
| PIXC0 Channel 0 DMA                     | 107 | 35          | LEVEL       |
| PIXC0 Channel 1 DMA                     | 108 | 36          | LEVEL       |
| PIXC0 Channel 2 DMA                     | 109 | 37          | LEVEL       |
| PIXC0 Status                            | 110 |             | LEVEL       |
| PVP0 Camera Pipe Data Out B DMA Channel | 111 | 38          | LEVEL       |
| PVP0 Camera Pipe Data Out C DMA Channel | 112 | 39          | LEVEL       |
| PVP0 Camera Pipe Status Out DMA Channel | 113 | 40          | LEVEL       |
| PVP0 Camera Pipe Control In DMA Channel | 114 | 41          | LEVEL       |



**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description                             | ID  | DMA Channel | Sensitivity |
|-----------------------------------------|-----|-------------|-------------|
| PVP0 Status 0                           | 115 |             | LEVEL       |
| PVP0 Memory Pipe Data Out DMA Channel   | 116 | 42          | LEVEL       |
| PVP0 Memory Pipe Data In DMA Channel    | 117 | 43          | LEVEL       |
| PVP0 Memory Pipe Status Out DMA Channel | 118 | 44          | LEVEL       |
| PVP0 Memory Pipe Control In DMA Channel | 119 | 45          | LEVEL       |
| PVP0 Camera Pipe Data Out A DMA Channel | 120 | 46          | LEVEL       |
| PVP0 Status 1                           | 121 |             | LEVEL       |
| USB0 Status/FIFO Data Ready             | 122 |             | LEVEL       |
| USB0 DMA Status/Transfer Complete       | 123 |             | LEVEL       |
| TRU0 Interrupt 0                        | 124 |             | PULSE/EDGE  |
| TRU0 Interrupt 1                        | 125 |             | PULSE/EDGE  |
| TRU0 Interrupt 2                        | 126 |             | PULSE/EDGE  |
| TRU0 Interrupt 3                        | 127 |             | PULSE/EDGE  |
| DMA Controller Error                    | 128 |             | LEVEL       |
| CGU0 Error                              | 129 |             | LEVEL       |
| Reserved                                | 130 |             |             |
| DPM0 Event                              | 131 |             | LEVEL       |
| Reserved                                | 132 |             |             |
| SWU0 Event                              | 133 |             | LEVEL       |

**Table 7-2:** ADSP-BF60x Interrupt List (Continued)

| Description | ID  | DMA Channel | Sensitivity |
|-------------|-----|-------------|-------------|
| SWU1 Event  | 134 |             | LEVEL       |
| SWU2 Event  | 135 |             | LEVEL       |
| SWU3 Event  | 136 |             | LEVEL       |
| SWU4 Event  | 137 |             | LEVEL       |
| SWU5 Event  | 138 |             | LEVEL       |
| SWU6 Event  | 139 |             | LEVEL       |

## **SEC Definitions**

The following definitions are used in describing the event controller.

### **System Events**

System source indications including interrupts and faults

### **System Source**

Point of origin of system event

### **SID (Identification, unique)**

Source numeric identifier for each system source connected to the SEC

### **SSI**

SEC Source Interface, system event source control and status sub-block of the SEC

### **SCI**

SEC Core Interface, core interface sub-block of the SEC

### **SPR**

SEC prioritizer determines the highest priority pending interrupt and the highest priority active interrupt and provides them in the appropriate registers of the SCI for the prioritization and nesting model of the SCI

### **SFI**

SEC Fault Interface, fault management sub-block of the SEC

## **SEC Block Diagram**

System sources connect to the SEC through the SSI. Each core has a dedicated SCI. The SFI provides fault action connections to the rest of the system.

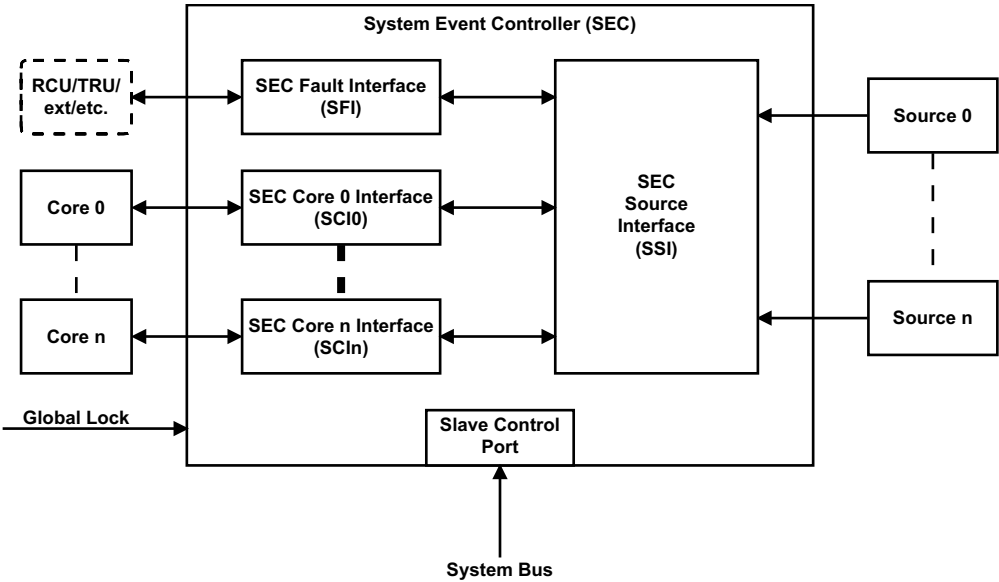


Figure 7-1: System Even Controller Block Diagram

SFI Block Diagram

The SFI manages fault events and associated actions. The fault management support provided in the SEC is intended to help satisfy the safety requirements of various applications. The SSI provides the highest priority pending source that is enabled as a fault. The SFI captures this value and enables a countdown and, once the countdown expires, the prescribed action is taken.

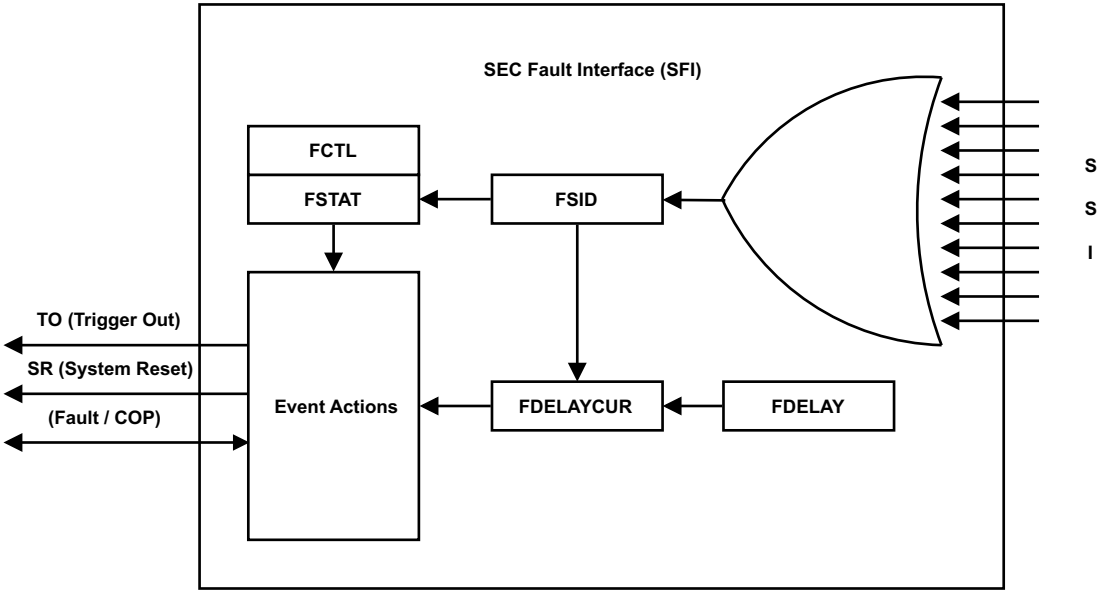


Figure 7-2: SFI Overview Block Diagram

## SCI Block Diagram

The SCI manages communication between the corresponding core and the SEC. The SPR of the SCI receives pending, active, and priority information from the SSI for each system event source assigned to this SCI. The SPR determines the highest priority pending system event and the SCI determines whether it will propagate to the core. The SCI maintains the coherency for the system event service model implemented on the connected core.

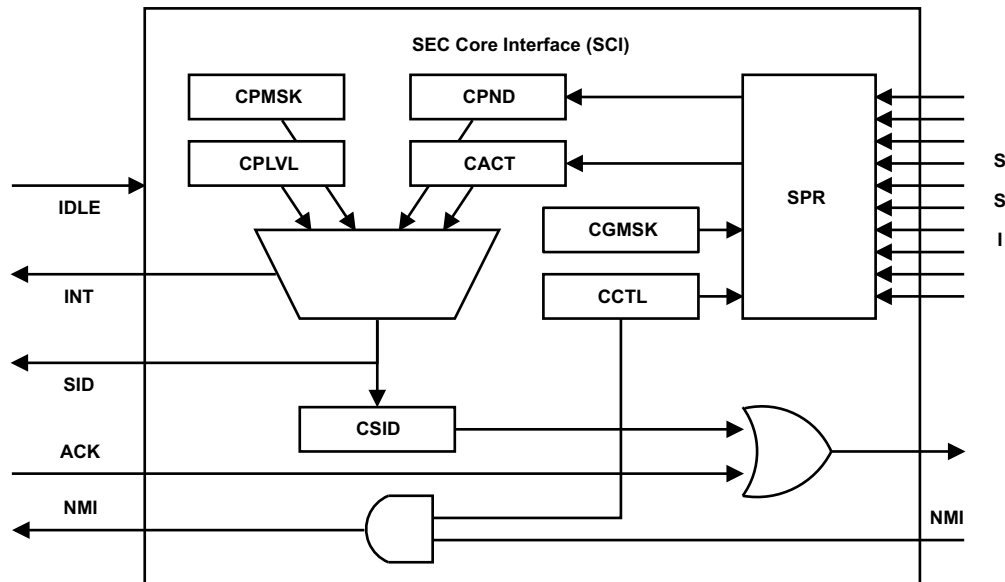


**Figure 7-3:** SCI Overview Block Diagram

## SSI Block Diagram

The SSI manages all of the system event sources. The status of each source is maintained in the corresponding SSTAT register and the control of each source is managed by the corresponding SCTL register. A pending and enabled event passes its indication and priority to the SCI to which it is assigned for further processing.

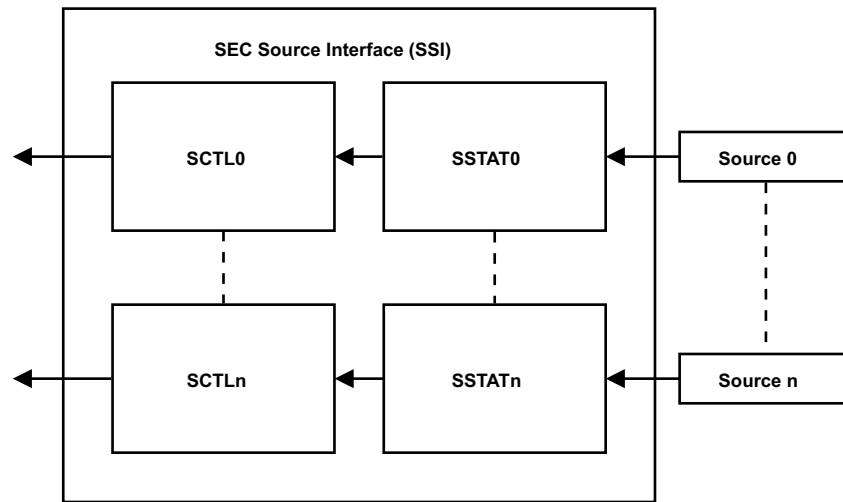


**Figure 7-4:** SSI Overview Block Diagram

## SEC Architectural Concepts

This section describes SEC architectural features supporting interrupt acknowledge, priority levels, grouping, flow, and error management.

### System Interrupt Acknowledge

System interrupt acknowledge is defined as the core providing indication that it has acquired the SID of the interrupt last issued by the SEC. The SEC core interface option allows for this to be generated by either of the following.

- Slave port write to the CSIDn register.
- Assertion of input ACK signal (generated by connected core)

### System Interrupt Groups

System sources can be assigned to groups using the SCTLn.GRP field. Source groups allow fast context switching for system interrupts at each SCI. The CGMSK register allows quick masking of unlimited size interrupt groups with a single write operation.

### System Interrupt Flow

An enabled and asserted system interrupt source is latched at the SSI and routed to the appropriate SCI based on the SCTLn.CTG bit field setting. The SEC prioritizer determines the highest priority pending system interrupt and the SCI updates the CPND register with the SID and PRIO bit field values. The SCI compares the CPND register value against the highest priority active source in the CACT register).

The `CPLVLM` register determines how many of the MSBs are used in the comparison while the `CPMSKm` and `CGMSK` registers determine which pending interrupt sources participate. If `CPND` register value is a higher priority (lower value) than that of the `CACT` register from the comparison based on `CPLVL` register, the system interrupt output is asserted and the `CSID` register is updated with the `SID` field of the `CPND` register and forwarded to the connected core.

After the core provides an interrupt acknowledgment (MMR write of the `CSID` register or the core version of the `CSID` register), the interrupt source is set active (`SCTL.ACT==1`) until the service is complete by a write of the same `SID` value to `END`.

#### Example Flow – Single Interrupt

1. The SEC compares the `CPND` register value to the `CACT` register value, if the interrupt in the `CPND` register is higher priority, continue.
2. The SEC copies the `CPND` register value to the `CSID` register and asserts the `INT` signal.
3. The core reads the `CSID` register (or core version).
4. The core writes to the `CSID` register (or core version, asserts the `ACK` signal).
5. The SEC de-asserts the `INT` signal and clears the `PND` bit and sets the `ACT` bit in the `SSTAT` register of the source going active.
6. The core writes the `SID` of the active interrupt to the `END` register.
7. The SEC clears the `ACT` bit in the `SSTAT` register of the source being ended.

#### Example Flow – Interrupt Nesting (Interrupt A is Lower Priority and Occurs Earlier than Interrupt B)

1. The SEC compares the `CPND` (A) register value to the `CACT` register and if the interrupt in the `CPND` register is a higher priority, continue.
2. The SEC copies `CPND` (A) register to the `CSID` register and asserts the `INT` signal.
3. The core reads the `CSID` (A) register (or core version).
4. The core writes to the `CSID` register (or core version, asserts the `ACK` signal).
5. The SEC de-asserts the `INT` signal and clears the `PND` bit and sets the `ACT` bit in the `SSTAT` register of the source (A) going active.
6. The SEC compares the `CPND` (B) register value to the `CACT` (A) register value. If the `CACT` (A) register value is a higher priority, continue.
7. The SEC copies the `CPND` (B) register value to `CSID` register and asserts the `INT` signal.
8. The core reads the `CSID` (B) register (or core version).
9. The core writes to the `CSID` register (or core version, asserts the `ACK` signal).

10. The SEC de-asserts the `INT` signal and clears the `PND` bit and sets the `ACT` bit in the `SSTAT` register of the source (B) going active.
11. The core writes the `SID` of the active interrupt (B) to the `END` register.
12. The SEC clears the `ACT` bit in the `SSTAT` register of the source (B) being ended.
13. The core writes the `SID` of the active interrupt (A) to the `END` register.
14. The SEC clears the `ACT` bit in the `SSTAT` register of the source (A) being ended.

## **SEC Error**

An SEC error (`GSTAT.ERR`) is included as a system source input to the SEC to allow for handling the error as an interrupt or fault.

## **SEC Programming Model**

Implementing a system interrupt service model using the SEC requires, at a minimum, proper configuration of a system interrupt source (for example a peripheral or DMA), core interrupt/event service model, and the SEC. The core must be configured for response to system interrupts from the SEC. The SEC must be configured to enable and map the system interrupt source to the correct SCI and to forward interrupts to the connected core.

The system interrupt source must be configured to generate interrupt assertions. Alternatively, software triggering may be used for interrupt assertion. Software driven interrupts are generated by writing the source ID of the interrupt to be triggered to the `RAISE` register.

## **Programming Concepts**

The following list provides the basic programming concepts necessary for configuring the system event controller.

- Configuring an SSI as a system interrupt for a specific core.
- Configuring an SCI to provide system interrupts to the connected core.
- Configuring an SSI as a system fault.
- Configuring the SFI to manage system faults.

## **Programming Examples**

This section provides example programming tasks that are typical for SEC usage.



## Configuring a System Source to Interrupt a Core

1. Write to the GCTL register to enable the SEC.
2. Write to the CCTL register of the specific SCI to enable interrupts to be sent to core.
3. Write to the SCTL register of specific source to enable the source as an interrupt and set the core target field to map the source to the appropriate SCI.

## Configuring a System Source as a Fault

1. Write to the GCTL register to enable the SEC.
2. Write to the FCTL register to configure specific fault actions.
3. Optionally write to the FDELAY bit field to specify fault delay.
4. Write to the SCTL register of a specific source to enable the source as a fault.

## SEC Register Descriptions

System Event Controller (SEC) contains the following registers.

**Table 7-3:** SEC Module Registers

| Name       | Description                   |
|------------|-------------------------------|
| SEC_CCTLn  | SCI Control Register n        |
| SEC_CSTATn | SCI Status Register n         |
| SEC_CPNDn  | Core Pending Register n       |
| SEC_CACTn  | SCI Active Register n         |
| SEC_CPMSKn | SCI Priority Mask Register n  |
| SEC_CGMSKn | SCI Group Mask Register n     |
| SEC_CPLVLn | SCI Priority Level Register n |
| SEC_CSIDn  | SCI Source ID Register n      |
| SEC_FCTL   | Fault Control Register        |
| SEC_FSTAT  | Fault Status Register         |

**Table 7-3:** SEC Module Registers (Continued)

| Name           | Description                               |
|----------------|-------------------------------------------|
| SEC_FSID       | Fault Source ID Register                  |
| SEC_FEND       | Fault End Register                        |
| SEC_FDLY       | Fault Delay Register                      |
| SEC_FDLY_CUR   | Fault Delay Current Register              |
| SEC_FSRDLY     | Fault System Reset Delay Register         |
| SEC_FSRDLY_CUR | Fault System Reset Delay Current Register |
| SEC_FCOPP      | Fault COP Period Register                 |
| SEC_FCOPP_CUR  | Fault COP Period Current Register         |
| SEC_GCTL       | Global Control Register                   |
| SEC_GSTAT      | Global Status Register                    |
| SEC_RAISE      | Global Raise Register                     |
| SEC_END        | Global End Register                       |
| SEC_SCTLn      | Source Control Register n                 |
| SEC_SSTATn     | Source Status Register n                  |

## SCI Control Register n

The SEC control register (SEC\_CCTLn) contains SCI control bits for all system sources.

**SEC\_CCTLn: SCI Control Register n - R/W**

Reset = 0x0000 0000

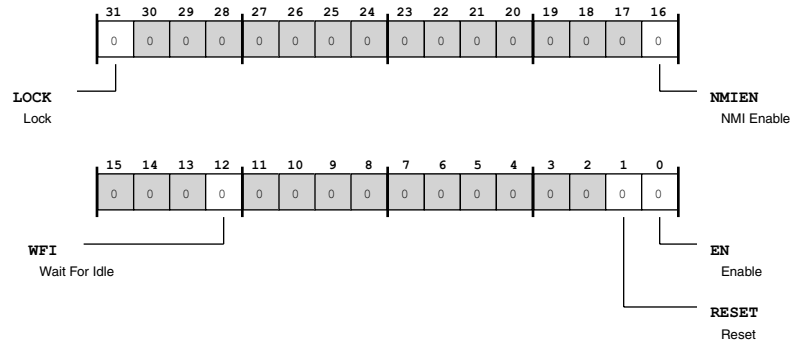


**Figure 7-5:** SEC\_CCTLn Register Diagram

**Table 7-4:** SEC\_CCTLn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_CCTLn.LOCK bit is enabled, the SEC_CCTLn register is read only.                                          |
|         |          | 0      Unlock                                                                                                                                                                     |
|         |          | 1      Lock                                                                                                                                                                       |
| 16      | NMIEN    | NMI Enable.<br>The SEC_CCTLn.NMIEN bit controls NMI propagation to the core. When the SEC_CCTLn.NMIEN bit is enabled, the SCI allows NMIs to propagate to the core for servicing. |
|         |          | 0      Disable                                                                                                                                                                    |
|         |          | 1      Enable                                                                                                                                                                     |
| 12      | WFI      | Wait For Idle.<br>When set, the SEC_CCTLn.WFI bit forces the SCI to wait for indication of core idle before the SCI resumes activity.                                             |
|         |          | 0      No Action                                                                                                                                                                  |
|         |          | 1      Wait for Idle                                                                                                                                                              |

**Table 7-4:** SEC\_CCTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | RESET    | Reset.<br>When set, the SEC_CCTLn.RESET bit resets all SCI registers to their default values.                                                                                                                                                                                        |
|         |          | 0                      No Action                                                                                                                                                                                                                                                     |
|         |          | 1                      Reset                                                                                                                                                                                                                                                         |
| 0       | EN       | Enable.<br>The SEC_CCTLn.EN bit controls operation of the SCI. Clearing the SEC_CCTLn.EN bit halts the execution of the SCI without resetting status registers. Setting the SEC_CCTLn.EN bit enables the SCI to begin or resume operation with the current configuration and status. |
|         |          | 0                      Disable                                                                                                                                                                                                                                                       |
|         |          | 1                      Enable                                                                                                                                                                                                                                                        |

SCI Status Register n

The SCI status register (SEC\_CSTATn) contains status bits, indicating the operational status of the SCI.

SEC\_CSTATn: SCI Status Register n - R/W

Reset = 0x0000 0000

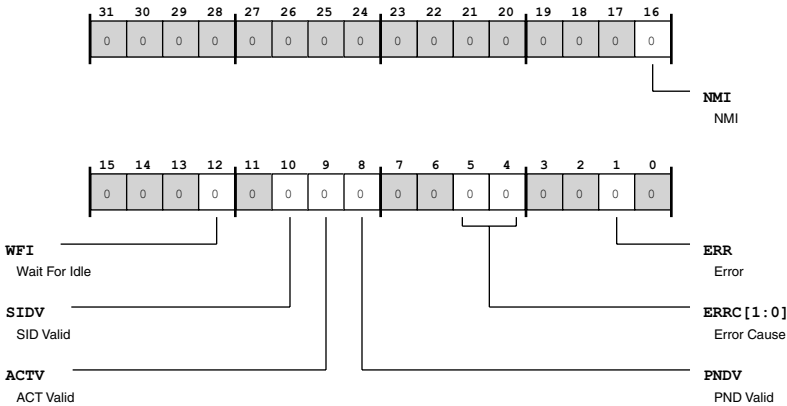


**Figure 7-6:** SEC\_CSTATn Register Diagram

**Table 7-5:** SEC\_CSTATn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | NMI      | NMI.<br>The SEC_CSTATn.NMI bit indicates whether an NMI has occurred since the bit was last cleared.                                                                                                                                                                                                                                                                                                       |
|         |          | 0                      No NMI Occured                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1                      NMI Occurred                                                                                                                                                                                                                                                                                                                                                                        |
| 12      | WFI      | Wait For Idle.<br>The SEC_CSTATn.WFI bit indicates (if set) that the SCI is temporarily disabled, pending a core idle indication. This bit is set when SEC_CCTLn.WFI is set.                                                                                                                                                                                                                               |
|         |          | 0                      Not Waiting                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1                      Waiting                                                                                                                                                                                                                                                                                                                                                                             |
| 10      | SIDV     | SID Valid.<br>The SEC_CSTATn.SIDV bit indicates (if set) that the current value in the SEC_CSIDn register is valid. The SCI sets the SEC_CSTATn.SIDV bit when the updating the SEC_CSIDn register with a new value. The SEC_CSTATn.SIDV bit is cleared when the SEC_CSIDn register is written. This status indication may be used to extract all pending interrupts in a single interrupt service routine. |
|         |          | 0                      Invalid                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Valid                                                                                                                                                                                                                                                                                                                                                                               |
| 9       | ACTV     | ACT Valid.<br>The SEC_CSTATn.ACTV bit indicates (if set) that the current value in the SEC_CACTn register is valid. The SCI sets the SEC_CSTATn.ACTV bit when updating the SEC_CACTn registers with a new value. The SEC_CSTATn.ACTV bit is cleared when the SEC_CSIDn register is written.                                                                                                                |
|         |          | 0                      Invalid                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Valid                                                                                                                                                                                                                                                                                                                                                                               |

**Table 7-5:** SEC\_CSTATn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                    |                                                                                                           |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 8       | PNDV     | PND Valid.<br>The SEC_CSTATn.PNDV bit indicates (if set) that the current value in the SEC_CPNDn register is valid. The SCI sets the SEC_CSTATn.PNDV bit when updating the SEC_CPNDn register with a new value. The SEC_CSTATn.PNDV bit is cleared when the SEC_CSIDn register is written. |                                                                                                           |
|         |          | 0                                                                                                                                                                                                                                                                                          | Invalid                                                                                                   |
|         |          | 1                                                                                                                                                                                                                                                                                          | Valid                                                                                                     |
| 5:4     | ERRC     | Error Cause.<br>The SEC_CSTATn.ERRC bits are updated on assertion of the SEC_CSTATn.ERR bit to indicate the SCI error type. SEC_CSTATn.ERRC is only updated on the assertion of SEC_CSTATn.ERR. Subsequent errors while SEC_CSTATn.ERR is asserted do not update SEC_CSTATn.ERRC.          |                                                                                                           |
|         |          | 0                                                                                                                                                                                                                                                                                          | Reserved                                                                                                  |
|         |          | 1                                                                                                                                                                                                                                                                                          | Acknowledge Error<br>SCI has received an acknowledge without a pending, unacknowledged interrupt present. |
|         |          | 2                                                                                                                                                                                                                                                                                          | Reserved                                                                                                  |
|         |          | 3                                                                                                                                                                                                                                                                                          | Reserved                                                                                                  |
|         |          |                                                                                                                                                                                                                                                                                            |                                                                                                           |
| 1       | ERR      | Error.<br>The SEC_CSTATn.ERR bit indicates that an error has occurred in the SCI. When SEC_CSTATn.ERR is set, the SCI updates the SEC_CSTATn.ERRC field to the value of the corresponding error cause.                                                                                     |                                                                                                           |
|         |          | 0                                                                                                                                                                                                                                                                                          | No Error                                                                                                  |
|         |          | 1                                                                                                                                                                                                                                                                                          | Error Occurred                                                                                            |

## Core Pending Register n

The SCI pending interrupt register (SEC\_CPNDn) contains the source ID and priority of the highest priority pending interrupt detected by the SEC prioritizer.

SEC\_CPNDn: Core Pending Register n - R/NW

Reset = 0x0000 0000

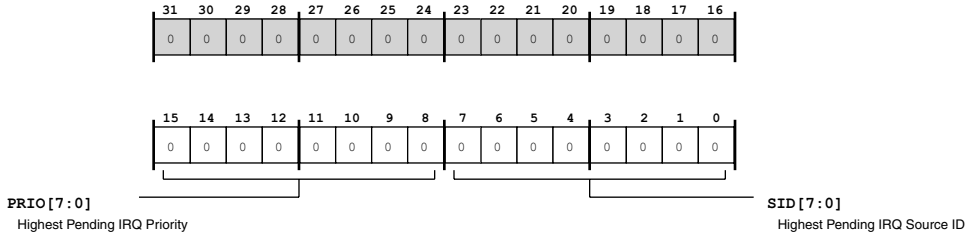


Figure 7-7: SEC\_CPNDn Register Diagram

Table 7-6: SEC\_CPNDn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | PRIO     | Highest Pending IRQ Priority.<br>The SEC_CPNDn.PRIO indicates the priority value of the highest priority pending interrupt for core n.   |
| 7:0     | SID      | Highest Pending IRQ Source ID.<br>The SEC_CPNDn.SID identifies the source ID value of the highest priority pending interrupt for core n. |

SCI Active Register n

The SEC SCI active interrupt register (SEC\_CACTn) contains the source ID and priority of the highest priority active interrupt detected by the SEC prioritizer.

SEC\_CACTn: SCI Active Register n - R/NW

Reset = 0x0000 0000

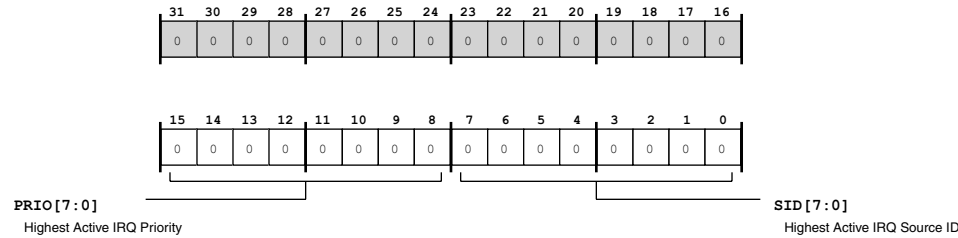


Figure 7-8: SEC\_CACTn Register Diagram

**Table 7-7:** SEC\_CACTn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | PRI0     | Highest Active IRQ Priority.<br>The SEC_CACTn.PRI0 indicates the priority value of the highest priority active interrupt for core n.   |
| 7:0     | SID      | Highest Active IRQ Source ID.<br>The SEC_CACTn.SID identifies the source ID value of the highest priority active interrupt for core n. |

SCI Priority Mask Register n

The SEC SCI priority mask register (SEC\_CPMSKn) contains the SCI priority mask for core n and includes a register lock.

SEC\_CPMSKn: SCI Priority Mask Register n - R/W

Reset = 0x0000 00ff

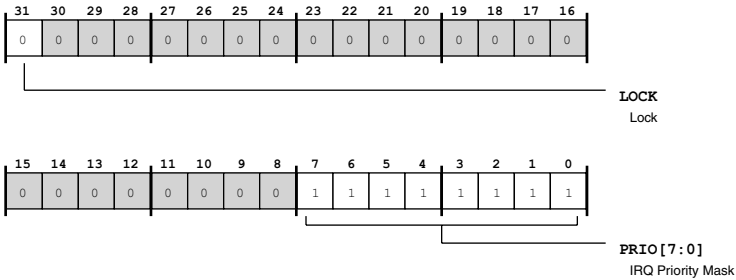


**Figure 7-9:** SEC\_CPMSKn Register Diagram

**Table 7-8:** SEC\_CPMSKn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_CPMSKn.LOCK bit is enabled, the SEC_CPMSKn register is read only. |
|         |          | 0                      Unlock                                                                                                              |
|         |          | 1                      Lock                                                                                                                |



**Table 7-8:** SEC\_CPMSK<sub>n</sub> Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | PRI0     | <p>IRQ Priority Mask.</p> <p>The SEC_CPMSK<sub>n</sub>.PRI0 contains the system interrupt priority mask for core n.</p> <p>0b11111111 = Priority Level 255 (lowest)</p> <p>...</p> <p>0b00000000 = Priority Level 0 (highest)</p> |

## SCI Group Mask Register n

The SEC SCI group mask register (SEC\_CGMSK<sub>n</sub>) contains selections for a group mask, an ungroup mask, and a register lock.

**SEC\_CGMSK<sub>n</sub>: SCI Group Mask Register n - R/W**

Reset = 0x0000 0000

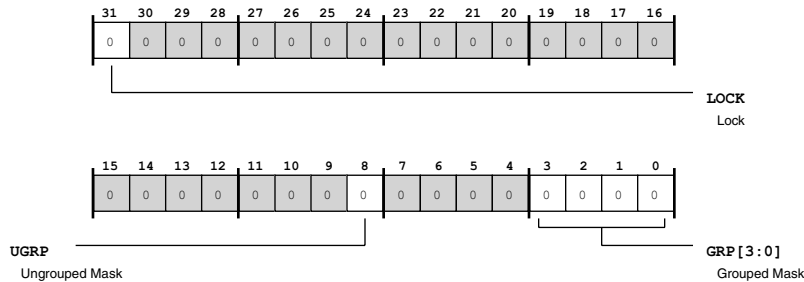


**Figure 7-10:** SEC\_CGMSK<sub>n</sub> Register Diagram

**Table 7-9:** SEC\_CGMSK<sub>n</sub> Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                      |        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------|--------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_CGMSKn . LOCK bit is enabled, the SEC_CGMSKn register is read only. |        |
|         |          | 0                                                                                                                                            | Unlock |
|         |          | 1                                                                                                                                            | Lock   |

**Table 7-9:** SEC\_CGMSK<sub>n</sub> Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | UGRP     | Ungrouped Mask.<br>The SEC_CGMSK <sub>n</sub> .UGRP bit masks interrupts (if set) for the ungrouped interrupt sources for core n.                                                                                                                                                                                  |
|         |          | 0                      Unmask Ungrouped Sources                                                                                                                                                                                                                                                                    |
|         |          | 1                      Mask Ungrouped Sources                                                                                                                                                                                                                                                                      |
| 3:0     | GRP      | Grouped Mask.<br>The SEC_CGMSK <sub>n</sub> .GRP field selects a group of interrupt sources to mask for core n. (For more information about interrupt source groups, see the SEC_SCTL <sub>n</sub> register description.)<br>1111 = Mask group 0, 1, 2, 3<br>...<br>0001 = Mask group 0<br>0000 = No groups masked |

SCI Priority Level Register n

The SEC SCI priority level register (SEC\_CPLVL<sub>n</sub>) contains selections for priority levels and a register lock.

SEC\_CPLVL<sub>n</sub>: SCI Priority Level Register n - R/W

Reset = 0x0000 0007

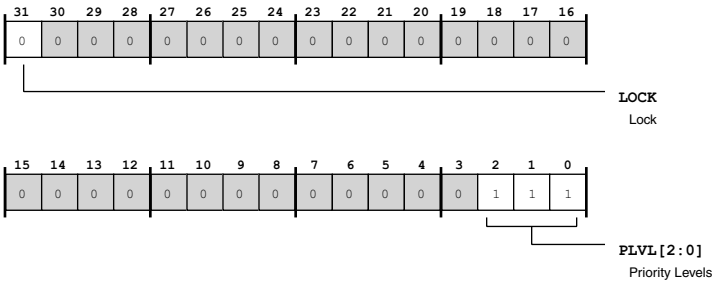


**Figure 7-11:** SEC\_CPLVL<sub>n</sub> Register Diagram

**Table 7-10:** SEC\_CPLVLn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_CPLVLn.LOCK bit is enabled, the SEC_CPLVLn register is read only.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0                      Unlock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1                      Lock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 2:0     | PLVL     | <p>Priority Levels.</p> <p>The SEC_CPLVLn.PLVL field serves to divide the total number of interrupt priority levels into sub-levels. The sub-level priority resolution provides the tie breaker for simultaneously pending interrupts assigned to the same interrupt level. The sub-level priority value specifies the number of MSBs (minus 1) designated to interrupt levels, while the remaining LSBs are designated for sub-level specification. For example, if the SEC_CPLVLn.PLVL field is set to two, the result is four priority levels are specified, because only the two MSBs are used for preemption evaluation. The remaining bits of the priority setting are used for sub-level prioritization.</p> <p>000 = 1 MSB (2 priority levels)</p> <p>...</p> <p>111 = 8 MSBs (256 priority levels)</p> |

## SCI Source ID Register n

The SCI source ID register (SEC\_CSIDn) contains the source ID of the interrupt last issued to core n. The SEC\_CSIDn register value is loaded by the SCI when a system interrupt indication is sent to core n. The SCI does not change the SEC\_CSIDn until after the interface receives an interrupt acknowledge from core n. Writing to the SEC\_CSIDn register generates an interrupt acknowledge, but does not update the value in the register.

SEC\_CSIDn: SCI Source ID Register n - R/W

Reset = 0x0000 0000

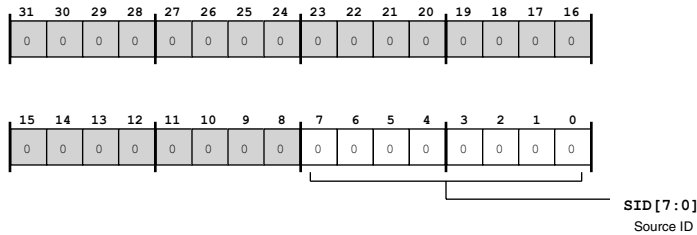


Figure 7-12: SEC\_CSIDn Register Diagram

Table 7-11: SEC\_CSIDn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | SID      | Source ID.              |

Fault Control Register

The SEC fault control register (SEC\_FCTL) contains fault control bits for all SEC channels.

SEC\_FCTL: Fault Control Register - R/W

Reset = 0x0000 0000

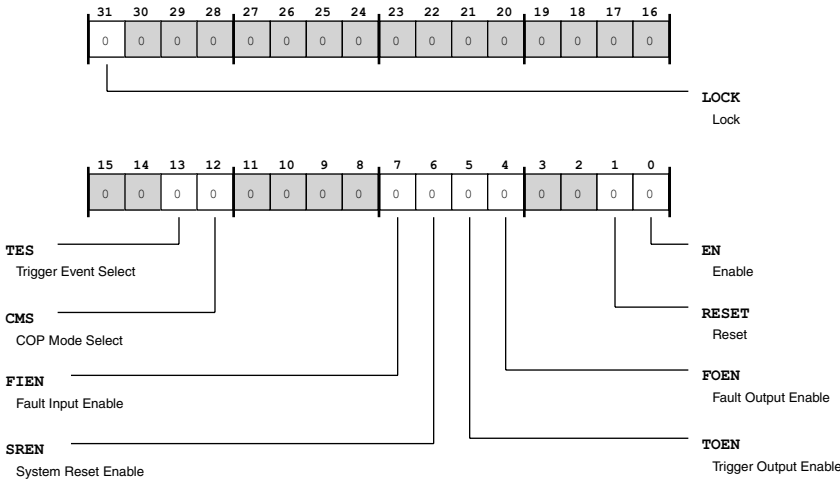


Figure 7-13: SEC\_FCTL Register Diagram

**Table 7-12:** SEC\_FCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_FCTL.LOCK bit is enabled, the SEC_FCTL register is read only.                                                                                                                                                                                                                                                                        |
|         |          | 0                      UnLock                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1                      Lock                                                                                                                                                                                                                                                                                                                                                                                   |
| 13      | TES      | Trigger Event Select.<br>The SEC_FCTL.TES bit selects the event that directs the SEC to assert trigger output. In fault pending mode, the SEC asserts trigger output when a fault is pending. In fault active mode, the SEC asserts trigger output when a fault is active.                                                                                                                                    |
|         |          | 0                      Fault Active Mode                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1                      Fault Pending Mode                                                                                                                                                                                                                                                                                                                                                                     |
| 12      | CMS      | COP Mode Select.<br>The SEC_FCTL.CMS selects the SEC mode for handling fault input. In COP mode, the SEC toggles the COP pin to indicate that no fault is active and ceases toggling the pin to indicate that a fault is active. In fault mode, the SEC deasserts the fault pin (=0) and fault_b pin (=1) when no fault is active and asserts the fault pin (=1) and fault_b pin (=0) when a fault is active. |
|         |          | 0                      Fault Mode                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      COP Mode                                                                                                                                                                                                                                                                                                                                                                               |
| 7       | FIEN     | Fault Input Enable.<br>The SEC_FCTL.FIEN bit enables the SEC the to sample fault input. If SEC_FCTL.FIEN is set (=1), a fault indication from an external device sets the SEC_FSTAT.ACT bit and SEC_FSID.FEXT bit.                                                                                                                                                                                            |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 7-12:** SEC\_FCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | SREN     | System Reset Enable.<br>The SEC_FCTL.SREN bit enables the SEC to issue a system reset request when a fault becomes active.                                                                                                                                                                    |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |
| 5       | TOEN     | Trigger Output Enable.<br>The SEC_FCTL.TOEN bit enables the SEC to produce trigger output when a fault becomes active.                                                                                                                                                                        |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |
| 4       | FOEN     | Fault Output Enable.<br>The SEC_FCTL.FOEN bit enables the SEC to indicate fault status, according to the SEC_FCTL.CMS bit configuration.                                                                                                                                                      |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |
| 1       | RESET    | Reset.<br>Setting the SEC_FCTL.RESET bit resets ALL SEC registers to their default values.                                                                                                                                                                                                    |
|         |          | 0      No Action                                                                                                                                                                                                                                                                              |
|         |          | 1      Reset                                                                                                                                                                                                                                                                                  |
| 0       | EN       | Enable.<br>The SEC_FCTL.EN bit controls the operational state of the SEC. Clearing the SEC_FCTL.EN bit halts the execution of the SEC without resetting status registers. Setting the SEC_FCTL.EN bit enables the SEC to begin or resume operation with the current configuration and status. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |

## Fault Status Register

The SEC fault status register (SEC\_FSTAT) contains status bits for all SEC channels.

**SEC\_FSTAT: Fault Status Register - R/W**

Reset = 0x0000 0000

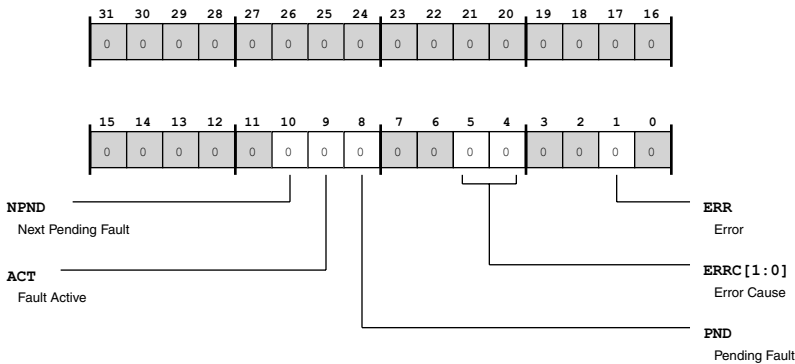


**Figure 7-14:** SEC\_FSTAT Register Diagram

**Table 7-13:** SEC\_FSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | NPND     | Next Pending Fault.<br>The SEC_FSTAT.NPND bit indicates that one or more sources have signalled fault assertion, but the input has not yet triggered the fault pending detection in the SEC fault interace. The SEC sets the SEC_FSTAT.NPND bit when the fault interface detects assertion of any enabled fault source input, while either SEC_FSTAT.PND or SEC_FSTAT.ACT bits are set. The SEC clears the SEC_FSTAT.NPND bit when there are no fault sources waiting. |
|         |          | 0                      Not Pending                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1                      Pending                                                                                                                                                                                                                                                                                                                                                                                                                                         |

**Table 7-13:** SEC\_FSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |              |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| 9       | ACT      | Fault Active.<br>The SEC_FSTAT.ACT bit indicates that the SEC has received a fault source input, the current fault delay count (in the SEC_FDLY_CUR register) has expired, and the fault actions are enabled. The SEC also sets the SEC_FSTAT.ACT bit on fault input detection if the SEC_FCTL.FIEN bit is set. The SEC_FSTAT.ACT bit is cleared by writing the SEC_END.SID value of the asserted fault from SEC_FSID register to the SEC_FEND register.                                                                                                                                                                                                                                                                                                                                        |              |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | No Fault     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Active Fault |
| 8       | PND      | Pending Fault.<br>The SEC_FSTAT.PND bit indicates a fault source has signalled a fault assertion to the SEC, but the SEC has not yet triggered the event actions due to delay selected with the SEC_FDLY register. The SEC fault interface sets the SEC_FSTAT.PND bit when the SEC_FSID is updated on assertion a fault source input. The SEC_FSTAT.PND bit is only set when the SEC_FSTAT.ACT bit is cleared. The SEC updates the SEC_FSID register with the SEC_END.SID value from the SEC_END register when the SEC_FSTAT.PND bit is set. The SEC_FSTAT.PND bit is cleared either by the SEC fault interface when the current delay count in the SEC_FDLY_CUR register expires or by writing the SEC_FSID.SID field value (indicates the ID of the asserted fault) to the SEC_FEND register. |              |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Not Pending  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Pending      |



**Table 7-13:** SEC\_FSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | ERRC     | <p>Error Cause.</p> <p>When the SEC updates the SEC_FSTAT.ERR bit, the SEC updates the SEC_FSTAT.ERRC bits to indicate the error type. When the error status is End Error, the status indicates two possible error scenerios. Either, the SEC received an end input was received from a fault source while neither the pending fault bit (SEC_FSTAT.PND) nor fault active bit (SEC_FSTAT.ACT) were set, or the SEC detected that the source ID value in the SEC_END.SID field does not match fault source indicated in the SEC_FEND.SID field.</p> |
|         |          | 0 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 2 End Error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 3 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 1       | ERR      | <p>Error.</p> <p>The SEC_FSTAT.ERR bit indicates an SEC fault interface error. When SEC_FSTAT.ERR is set, the SEC updates the SEC_FSTAT.ERRC field to indicated the corresponding error cause. When multiple errors occur, the SEC_FSTAT register captures the status for the first error and does not capture subsequent errors until the status is cleared.</p>                                                                                                                                                                                  |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Fault Source ID Register

The SEC fault source ID register (SEC\_FSID) contains a fault source ID and internal/external fields.

SEC\_FSID: Fault Source ID Register - R/NW

Reset = 0x0000 0000

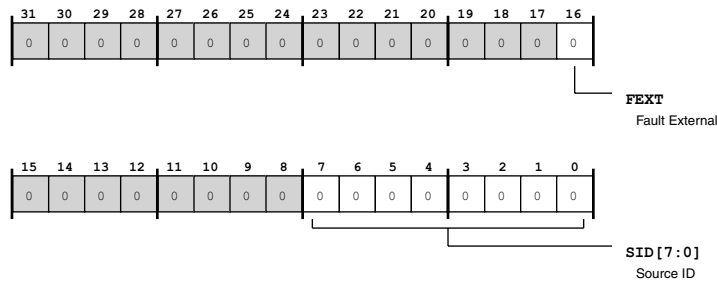


Figure 7-15: SEC\_FSID Register Diagram

Table 7-14: SEC\_FSID Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | FEXT     | Fault External.<br>The SEC_FSID.FEXT bit indicates that the last active fault was asserted by an external device. The SEC sets the SEC_FSID.FEXT bit when the SEC_FSTAT register's SEC_FSTAT.ACT bit is set by the fault input pins. The SEC_FSID.FEXT bit is cleared when the SEC_FSTAT.ACT bit is set by an internal fault or when the external fault is ended. When the SEC_FSID.FEXT bit is set, the SEC_FSID.SID is cleared. |
|         |          | 0      Fault Internal                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1      Fault External                                                                                                                                                                                                                                                                                                                                                                                                             |
| 7:0     | SID      | Source ID.<br>The SEC_FSID.SID identifies the fault assertion detected by the SEC fault interface. The SEC loads the SEC_FSID.SID field value when a system fault indication is asserted. The SEC fault interface does not change the SEC_FSID.SID value until the fault is no longer pending or active, as indicated by the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit being cleared in the SEC_FSTAT register.                     |

Fault End Register

The SEC fault end register (SEC\_FEND) contains fault source ID and internal/external fields.

SEC\_FEND: Fault End Register - R/W

Reset = 0x0000 0000

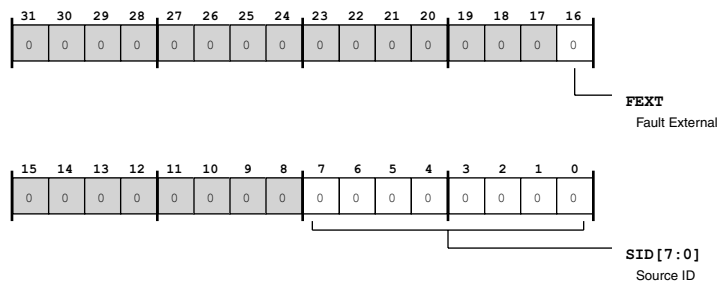


Figure 7-16: SEC\_FEND Register Diagram

Table 7-15: SEC\_FEND Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | FEXT     | Fault External.<br>Setting the SEC_FEND.FEXT bit, when the SEC_FEND.SID field is cleared, clears an active fault from an external source.                                                                                                                          |
|         |          | 0      Fault Internal                                                                                                                                                                                                                                              |
|         |          | 1      Fault External                                                                                                                                                                                                                                              |
| 7:0     | SID      | Source ID.<br>The SEC_FEND.SID identifies a fault to be ended as indicated to the SEC by the core. The core loads the SEC_FEND.SID field value. If the SEC_FEND.SID value matches the SEC_FSID.SID value, the SEC_FSTAT.PND bit and SEC_FSTAT.ACT bit are cleared. |

Fault Delay Register

The SEC fault delay register (SEC\_FDLY) contains the number (SEC\_FDLY.COUNT field) of SCLK periods to delay from fault pending to fault active, when actions are enabled.

SEC\_FDLY: Fault Delay Register - R/W

Reset = 0x0000 0000

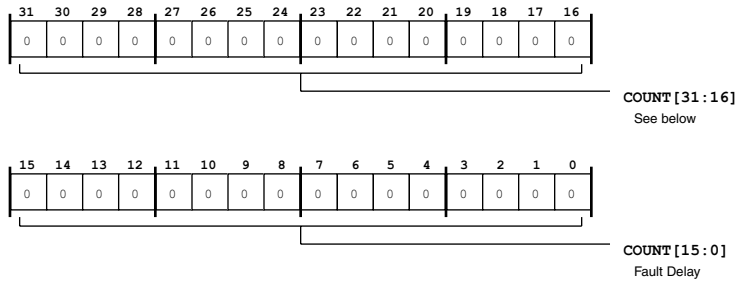


Figure 7-17: SEC\_FDLY Register Diagram

Table 7-16: SEC\_FDLY Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | COUNT    | Fault Delay.            |

Fault Delay Current Register

The SEC fault delay current register (SEC\_FDLY\_CUR) contains the active count (SEC\_FDLY\_CUR.COUNT field) in SCLK periods for the delay from fault pending to fault active, when actions are enabled. The count is loaded from the SEC\_FDLY register when a fault becomes pending (SEC\_FSTAT.PND bit is set). The SEC decrements the value in SEC\_FDLY\_CUR each SCLK cycle while the SEC\_FSTAT.PND bit is set.

SEC\_FDLY\_CUR: Fault Delay Current Register - R/NW

Reset = 0x0000 0000

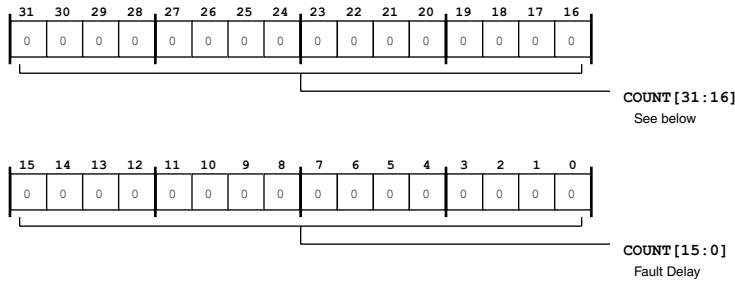


Figure 7-18: SEC\_FDLY\_CUR Register Diagram

**Table 7-17:** SEC\_FDLY\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | COUNT    | Fault Delay.            |

## Fault System Reset Delay Register

The SEC fault system reset delay register (SEC\_FSRDLY) contains the number (SEC\_FSRDLY.COUNT field) of SCLK periods for the delay from a fault becoming active to system reset request assertion, if enabled.

**SEC\_FSRDLY: Fault System Reset Delay Register - R/W**

Reset = 0x0000 0000

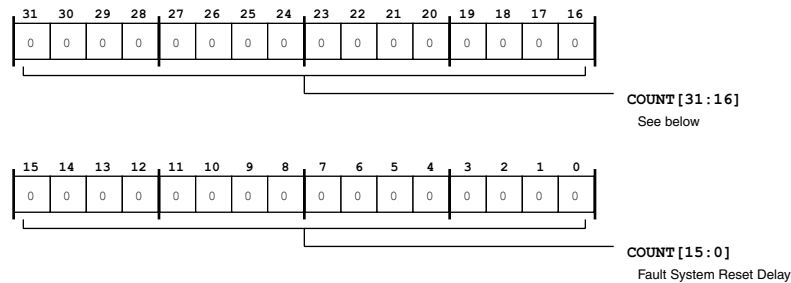


**Figure 7-19:** SEC\_FSRDLY Register Diagram

**Table 7-18:** SEC\_FSRDLY Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | COUNT    | Fault System Reset Delay. |

## Fault System Reset Delay Current Register

The SEC fault system reset delay current register (SEC\_FSRDLY\_CUR) contains the active count (SEC\_FSRDLY\_CUR.COUNT field) in SCLK periods for the delay from fault active to system reset assertion, if enabled. The count is loaded from the SEC\_FSRDLY register when a fault becomes active (SEC\_FSTAT.ACT bit is set). The SEC decrements the value in SEC\_FSRDLY\_CUR each SCLK cycle while the SEC\_FSTAT.ACT bit is set.

SEC\_FSRDLY\_CUR: Fault System Reset Delay Current Register - R/NW

Reset = 0x0000 0000

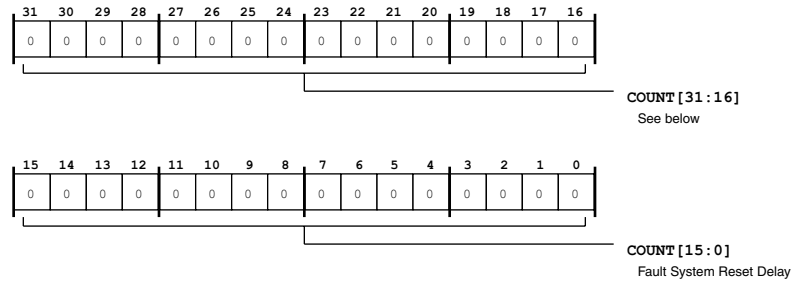


Figure 7-20: SEC\_FSRDLY\_CUR Register Diagram

Table 7-19: SEC\_FSRDLY\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | COUNT    | Fault System Reset Delay. |

Fault COP Period Register

The SEC fault COP period register (SEC\_FCOPP) contains the width value (count in SCLK cycles) for the high and low phase of the computer operating properly (COP) toggled output on the COP pin. Note that the actual high/low phase is value is the SEC\_FCOPP.COUNT programmed value plus 1.

SEC\_FCOPP: Fault COP Period Register - R/W

Reset = 0x0000 0000

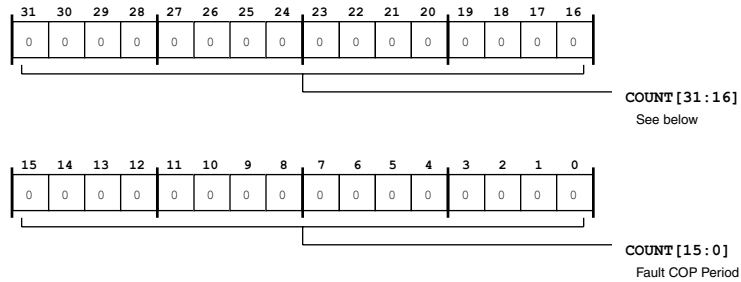


Figure 7-21: SEC\_FCOPP Register Diagram

Table 7-20: SEC\_FCOPP Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | COUNT    | Fault COP Period.       |

## Fault COP Period Current Register

The SEC fault COP period current register (SEC\_FCOPP\_CUR) contains the active count (in SCLK periods) for the current phase (high or low) of the computer operating properly (COP) toggled output on the COP pin. The SEC loads the SEC\_FCOPP\_CUR register from the SEC\_FCOPP register when the SEC\_FCOPP\_CUR.COUNT field is cleared and the SEC is in COP mode (SEC\_FCTL.CMS bit =1). The SEC decrements the SEC\_FCOPP\_CUR count each SCLK cycle while SEC\_FCTL.CMS is set and the SEC\_FSTAT.ACT bit is not set.

SEC\_FCOPP\_CUR: Fault COP Period Current Register - R/NW

Reset = 0x0000 0000

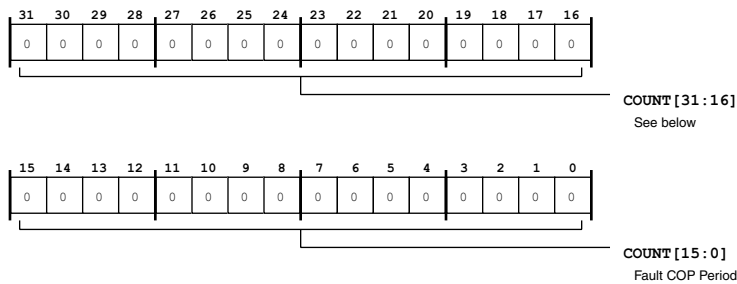


Figure 7-22: SEC\_FCOPP\_CUR Register Diagram

Table 7-21: SEC\_FCOPP\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | COUNT    | Fault COP Period.       |

## Global Control Register

The SEC global control register (SEC\_GCTL) provides register locking, reset, and, and enable for the SEC module.

**SEC\_GCTL: Global Control Register - R/W**

Reset = 0x0000 0000

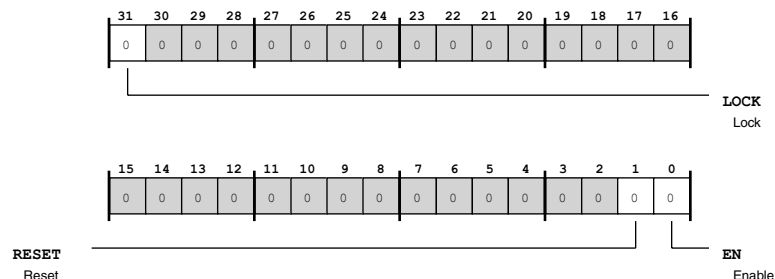


**Figure 7-23:** SEC\_GCTL Register Diagram

**Table 7-22:** SEC\_GCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_GCTL . LOCK bit is enabled, the SEC_GCTL register is read only.                                                                                                                                                                                                                                            |
|         |          | 0      Unlock                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Lock                                                                                                                                                                                                                                                                                                                                                                         |
| 1       | RESET    | Reset.<br>The SEC_GCTL . RESET bit is write-1-action and triggers a soft reset to all SEC registers.                                                                                                                                                                                                                                                                                |
|         |          | 0      No Action                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1      Reset                                                                                                                                                                                                                                                                                                                                                                        |
| 0       | EN       | Enable.<br>The SEC_GCTL . EN bit is read/write and must be set for the SEC to begin/resume SEC operation with the current configuration and status. Clearing SEC_GCTL . EN bit halts the execution of the SEC core interface (SCI). All SEC fault interfaces (SFI) and SEC source interfaces (SSI) remain active along with all error detection without resetting status registers. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                       |



## Global Status Register

The SEC global status register (SEC\_GSTAT) contains status bits for all SEC channels.

### SEC\_GSTAT: Global Status Register - R/W

Reset = 0x0000 0000

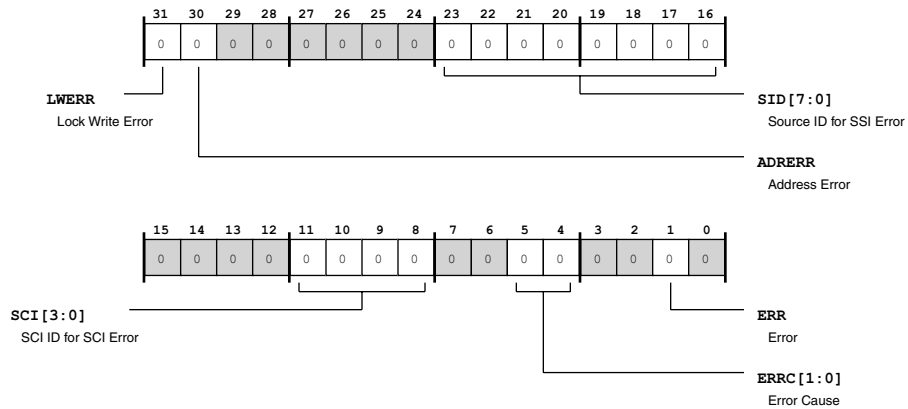


**Figure 7-24:** SEC\_GSTAT Register Diagram

**Table 7-23:** SEC\_GSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LWERR    | Lock Write Error.<br>The SEC_GSTAT.LWERR bit indicates (when set) there was an attempted write to an SEC register while the SEC_GCTL.LOCK bit was set and while the global lock bit was enabled (SPU_CTL_GLCK bit =1). This status bit is sticky; write-1-to-clear it. |
|         |          | 0 No Error                                                                                                                                                                                                                                                             |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                       |
| 30      | ADRERR   | Address Error.<br>The SEC_GSTAT.ADRERR bit indicates that the SEC generated and address error. This status bit is sticky; write-1-to-clear it.                                                                                                                         |
|         |          | 0 No Error                                                                                                                                                                                                                                                             |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                       |
| 23:16   | SID      | Source ID for SSI Error.<br>The SEC_GSTAT.SID bits indicate the source ID that generated the last SSI Error conveyed in the SEC_GSTAT.ERRC field.                                                                                                                      |

**Table 7-23:** SEC\_GSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:8    | SCI      | SCI ID for SCI Error.<br>The SEC_GSTAT.SCI bits indicate the number for the specific SCI that generated the last SCI error conveyed in the SEC_GSTAT.ERRC field.                                                                                                                                                                                                              |
| 5:4     | ERRC     | Error Cause.<br>When the SEC updates the SEC_GSTAT.ERR bit, the SEC updates the SEC_GSTAT.ERRC bits to indicate the error type. Note that for SCI errors the error status represents an OR of all the errors from each SCI. Note that for SSI errors the error status indicates an error is active for any SSI input. This error is an OR of all the interrupt source errors. |
|         |          | 0 SFI Error                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 SCI Error                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 2 SSI Error                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 3 Reserved                                                                                                                                                                                                                                                                                                                                                                    |
| 1       | ERR      | Error.<br>The SEC_GSTAT.ERR bit indicates an error has occurred in the SEC. When the SEC asserts this bit (=1), the SEC updates the SEC_GSTAT.ERRC field to indicate the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of this bit. This status bit is sticky; write-1-to-clear it.                                 |
|         |          | 0 No Error                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                                                                                                              |

## Global Raise Register

The SEC global raise register (SEC\_RAISE) contains a source ID interrupt set to pending field (SEC\_RAISE.SID). When a source ID value is written to this field, the SEC raises the source's interrupt to pending.

SEC\_RAISE: Global Raise Register - R/W

Reset = 0x0000 0000

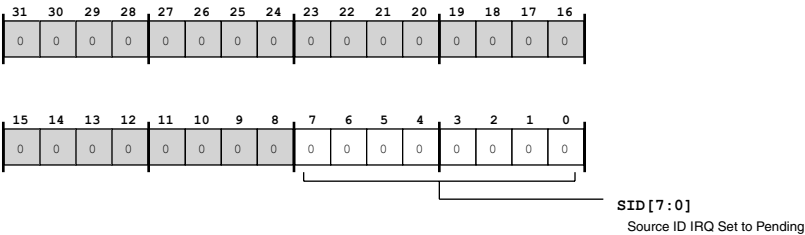


Figure 7-25: SEC\_RAISE Register Diagram

Table 7-24: SEC\_RAISE Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 7:0     | SID      | Source ID IRQ Set to Pending. |

Global End Register

The SEC global end register (SEC\_END) contains a source ID interrupt end service field (SEC\_END.SID). When a core has finished servicing an interrupt, the core writes the SEC\_END.SID field in the SEC\_END register. This write causes the SEC to clear the SEC\_SSTATn.ACT bit in the SEC\_SSTATn register of the corresponding interrupt.

SEC\_END: Global End Register - R/W

Reset = 0x0000 0000

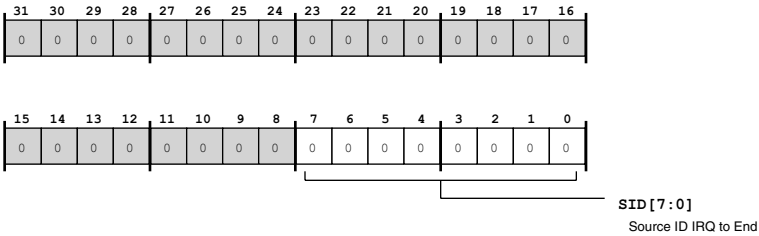


Figure 7-26: SEC\_END Register Diagram

Table 7-25: SEC\_END Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | SID      | Source ID IRQ to End.   |

Source Control Register n

The SEC interrupt source control register (SEC\_SCTLn) contains control bits to configure the SEC interrupt sources for core n.

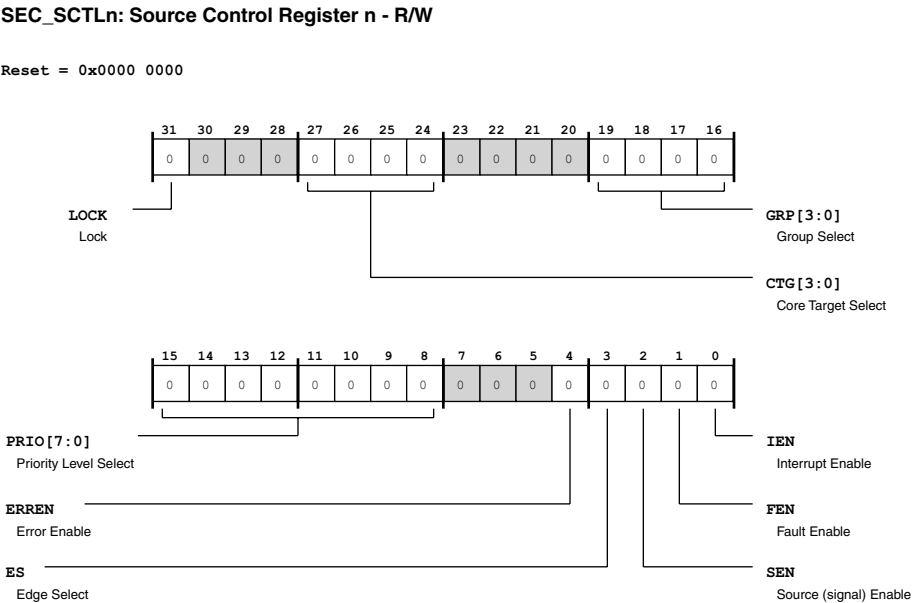


Figure 7-27: SEC\_SCTLn Register Diagram

Table 7-26: SEC\_SCTLn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the SEC_SCTLn.LOCK bit is enabled, the SEC_SCTLn register is read only.                                                                                                  |
|         |          | 0                      Unlock                                                                                                                                                                                                             |
|         |          | 1                      Lock                                                                                                                                                                                                               |
| 27:24   | CTG      | Core Target Select.<br>The SEC_SCTLn.CTG bits selects the specific SEC core interface to which the interrupt is mapped. Each system interrupt is mapped uniquely to one specific SEC core interface and (as a result) to a specific core. |

**Table 7-26:** SEC\_SCTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 19:16   | GRP      | <p>Group Select.</p> <p>The <code>SEC_SCTLn.GRP</code> bits each select a specific group for the interrupt. Each system interrupt can be assigned to any combination of groups supported by the <code>SEC_SCTLn.GRP</code> field.</p> <p>For example, consider the situation where <code>SEC_SCTLn.GRP0</code> represents interrupt group 0, <code>SEC_SCTLn.GRP1</code> represents interrupt group 1, and so on. One group might be used for all enabled interrupts (for example, group 0) and an additional group might be used for all wakeup interrupts (for example, group 1). This approach supports a model of all interrupts and just the wakeup subset.</p> <p>Before going to idle or sleep, all non-wakeup interrupts can be masked off to allow only wakeup interrupts to be enabled for service. Selecting no group (all <code>SEC_SCTLn.GRP</code> bits = 0) places the interrupt source in the category of "ungrouped".</p> |                 |
| 15:8    | PRI0     | <p>Priority Level Select.</p> <p>The <code>SEC_SCTLn.PRI0</code> bits sets the relative priority for an interrupt request. A pending interrupt request forwards its <code>SEC_SCTLn.PRI0</code> value to the SEC core interface.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                 |
| 4       | ERREN    | <p>Error Enable.</p> <p>The <code>SEC_SCTLn.ERREN</code> bit permits the <code>SEC_SSTATn.ERR</code> status bit to be set on error detection. If <code>SEC_SCTLn.ERREN</code> is cleared, no errors are detected.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Disable         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Enable          |
| 3       | ES       | <p>Edge Select.</p> <p>The <code>SEC_SCTLn.ES</code> bit selects the operational and sensitivity mode of the SEC source interface input.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Level Sensitive |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Edge Sensitive  |

**Table 7-26:** SEC\_SCTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                           |         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 2       | SEN      | <p>Source (signal) Enable.</p> <p>The SEC_SCTLn.SEN bit controls whether the system interrupt source input signal may affect the pending status of the source. Clearing SEC_SCTLn.SEN disables the source input signal from affecting pending. Setting SEC_SCTLn.SEN enables the source input signal to affect pending.</p>                                       |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                 | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                 | Enable  |
| 1       | FEN      | <p>Fault Enable.</p> <p>The SEC_SCTLn.FEN bit controls whether the SEC may forward an interrupt request to the SEC fault interface as a fault source. This bit does not affect the ability of an interrupt source to set an interrupt as pending. The SEC_SCTLn.FEN bit only affects whether the pending request may be forwarded to the SEC fault interface.</p> |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                 | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                 | Enable  |
| 0       | IEN      | <p>Interrupt Enable.</p> <p>The SEC_SCTLn.IEN bit controls whether the SEC may forward an interrupt request to a core for servicing. This bit does not affect the ability of an interrupt source to set an interrupt as pending.</p>                                                                                                                              |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                 | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                 | Enable  |

## Source Status Register n

The SEC interrupt source status register (SEC\_SSTATn) contains bits indicating the status of the corresponding interrupt source n. An interrupt source may be: pending, active, active and pending, or neither pending nor active.

**SEC\_SSTATn: Source Status Register n - R/W**

Reset = 0x0000 0000

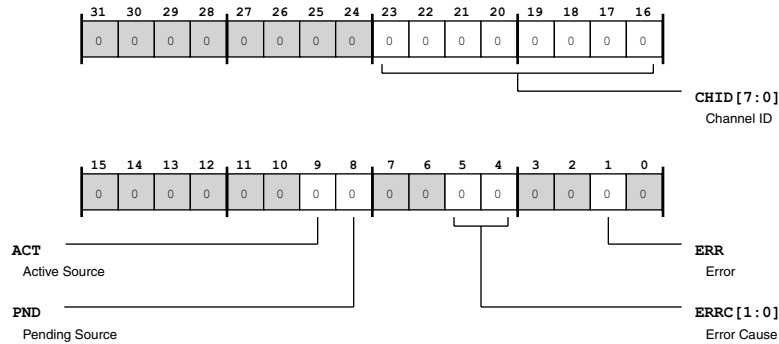


**Figure 7-28:** SEC\_SSTATn Register Diagram

**Table 7-27:** SEC\_SSTATn Register Fields

| Bit No. | Bit Name      | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |   |           |   |               |
|---------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|-----------|---|---------------|
| 23:16   | CHID          | <p>Channel ID.</p> <p>The SEC_SSTATn.CHID bits indicate the ID of the specific source (from a set of sources sharing one SEC source interface input) that asserted the SEC source interface input. An SEC source interface input may support multiple system sources, in which case the assertion must be qualified by an identifier to determine the channel that generated the assertion. The SEC_SSTATn.CHID field provides this value in the form of a numeric reference that is mapped to a specific interrupt source. The prioritization for simultaneously asserted sources is according to ID, with 0 being the highest priority. The SEC_SSTATn.CHID is captured when the SEC source interface input is acknowledged.</p> |   |           |   |               |
| 9       | ACT           | <p>Active Source.</p> <p>The SEC_SSTATn.ACT bit indicates the source has been accepted by a core for servicing, but the service is not yet complete. An SEC_SSTATn.ACT bit is set by the SEC when the specific system interrupt is acknowledged by the core through the SEC core interface. An SEC_SSTATn.ACT bit is cleared by the SEC when the core provides interrupt service end indication for the specific system interrupt through the SEC core interface.</p> <table><tr><td>0</td><td>No Source</td></tr><tr><td>1</td><td>Active Source</td></tr></table>                                                                                                                                                                | 0 | No Source | 1 | Active Source |
| 0       | No Source     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |   |           |   |               |
| 1       | Active Source |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |   |           |   |               |

**Table 7-27:** SEC\_SSTATn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                       |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 8       | PND      | Pending Source.<br>The SEC_SSTATn.PND bit indicates the source has signalled an interrupt request assertion, but the request has not yet been accepted by a core for servicing. A SEC_SSTATn.PND bit is set by the SEC on detection of an assertion of the corresponding system interrupt input. A SEC_SSTATn.PND bit is cleared by the SEC when the specific system interrupt is acknowledged by the core through the SEC core interface.                                                                                         |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Not Pending           |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Pending               |
| 5:4     | ERRC     | Error Cause.<br>When the SEC_SSTATn.ERR bit is asserted, the SEC updates SEC_SSTATn.ERRC field to convey the interrupt source error type. When the error type is source overflow, the status indicates that a source signal assertion occurred or an SEC raise operation was attempted while pending was already set. The source overflow is detected when the source is set for edge only. When the error type is end error, the status indicates that an end was received for a source while the SEC_SSTATn.ACT bit was not set. |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Source Overflow Error |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Reserved              |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | End Error             |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Reserved              |
| 1       | ERR      | Error.<br>The SEC_SSTATn.ERR bit indicates an error for a specific system interrupt source. When the SEC_SSTATn.ERR bit is set, the SEC updates the SEC_SSTATn.ERRC field to the value of the corresponding error cause. Even if multiple errors occur, only the first error is captured on assertion of the SEC_SSTATn.ERR bit.                                                                                                                                                                                                   |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | No Error              |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Error Occurred        |







## 8 Trigger Routing Unit (TRU)

The TRU provides system-level sequence control without core intervention. The TRU maps trigger masters (generators of triggers) to trigger slaves (receivers of triggers). Slave endpoints can be configured to respond to triggers in various ways. Common applications enabled by the TRU include:

- Automatically triggering the start of a DMA sequence after a sequence from another DMA channel completes
- Software triggering
- Synchronization of concurrent activities

### TRU Features

The TRU supports the following features.

- Trigger routing of any trigger master to any trigger slave.
- Software generation of any trigger master ID.
- Configuration protection through register level lock bits and global lock indication.

### TRU Functional Description

The following sections provide a function description of the TRU.

### TRU Register List

The trigger routing unit (TRU) provides simple sequence control of distributed modules without the penalties associated with core intervention (for example, interrupt overhead). The TRU resides in the SYSCCLK domain and receives trigger inputs from all master trigger inputs (MTI) and the TRU master trigger register (TRU\_MTR). Based on these inputs, the TRU logic generates trigger outputs that initiate slave operations in the processor core and peripherals. A set of registers govern TRU operations. For more information on TRU functionality, see the TRU register descriptions.

**Table 8-1:** TRU Module Registers

| Name        | Description                 |
|-------------|-----------------------------|
| TRU_SSRn    | Slave Select Register       |
| TRU_MTR     | Master Trigger Register     |
| TRU_ERRADDR | Error Address Register      |
| TRU_STAT    | Status Information Register |
| TRU_REVID   | Revision ID Register        |
| TRU_GCTL    | Global Control Register     |

## ADSP-BF60x Trigger List

**Table 8-2:** Masters ADSP-BF60x Trigger List

| Description               | ID | Sensitivity |
|---------------------------|----|-------------|
| Reserved                  | 0  |             |
| CGU0 Event                | 1  | PULSE/EDGE  |
| TIMER0 Timer 0            | 2  | PULSE/EDGE  |
| TIMER0 Timer 1            | 3  | PULSE/EDGE  |
| TIMER0 Timer 2            | 4  | PULSE/EDGE  |
| TIMER0 Timer 3            | 5  | PULSE/EDGE  |
| TIMER0 Timer 4            | 6  | PULSE/EDGE  |
| TIMER0 Timer 5            | 7  | PULSE/EDGE  |
| TIMER0 Timer 6            | 8  | PULSE/EDGE  |
| TIMER0 Timer 7            | 9  | PULSE/EDGE  |
| PINT0 Pin Interrupt Block | 10 | LEVEL       |
| PINT1 Pin Interrupt Block | 11 | LEVEL       |

**Table 8-2:** Masters ADSP-BF60x Trigger List (Continued)

| Description               | ID | Sensitivity |
|---------------------------|----|-------------|
| PINT2 Pin Interrupt Block | 12 | LEVEL       |
| PINT3 Pin Interrupt Block | 13 | LEVEL       |
| PINT4 Pin Interrupt Block | 14 | LEVEL       |
| PINT5 Pin Interrupt Block | 15 | LEVEL       |
| CNT0 Status               | 16 | LEVEL       |
| PWM0 PWMTMR Group         | 17 | LEVEL       |
| PWM1 PWMTMR Group         | 18 | LEVEL       |
| ACM0 Event Complete       | 19 | LEVEL       |
| SPORT0 Channel A DMA      | 20 | PULSE/EDGE  |
| SPORT0 Channel B DMA      | 21 | PULSE/EDGE  |
| SPORT1 Channel A DMA      | 22 | PULSE/EDGE  |
| SPORT1 Channel B DMA      | 23 | PULSE/EDGE  |
| SPORT2 Channel A DMA      | 24 | PULSE/EDGE  |
| SPORT2 Channel B DMA      | 25 | PULSE/EDGE  |
| SPI0 TX DMA Channel       | 26 | PULSE/EDGE  |
| SPI0 RX DMA Channel       | 27 | PULSE/EDGE  |
| SPI1 TX DMA Channel       | 28 | PULSE/EDGE  |
| SPI1 RX DMA Channel       | 29 | PULSE/EDGE  |
| RSI0 DMA Channel          | 30 | PULSE/EDGE  |
| SDU0 DMA                  | 31 | PULSE/EDGE  |
| Reserved                  | 32 |             |
| EMAC0 Status              | 33 | LEVEL       |

**Table 8-2:** Masters ADSP-BF60x Trigger List (Continued)

| Description                                                 | ID | Sensitivity |
|-------------------------------------------------------------|----|-------------|
| EMAC1 Status                                                | 34 | LEVEL       |
| LP0 DMA Channel                                             | 35 | PULSE/EDGE  |
| LP1 DMA Channel                                             | 36 | PULSE/EDGE  |
| LP2 DMA Channel                                             | 37 | PULSE/EDGE  |
| LP3 DMA Channel                                             | 38 | PULSE/EDGE  |
| UART0 Transmit DMA                                          | 39 | PULSE/EDGE  |
| UART0 Receive DMA                                           | 40 | PULSE/EDGE  |
| UART1 Transmit DMA                                          | 41 | PULSE/EDGE  |
| UART1 Receive DMA                                           | 42 | PULSE/EDGE  |
| Memory DMA Stream 0 Source /<br>CRC0 Input Channel          | 43 | PULSE/EDGE  |
| Memory DMA Stream 0<br>Destination / CRC0 Output<br>Channel | 44 | PULSE/EDGE  |
| Memory DMA Stream 1 Source /<br>CRC1 Input Channel          | 45 | PULSE/EDGE  |
| Memory DMA Stream 1<br>Destination / CRC1 Output<br>Channel | 46 | PULSE/EDGE  |
| Memory DMA Stream 2 Source<br>Channel                       | 47 | PULSE/EDGE  |
| Memory DMA Stream 2<br>Destination Channel                  | 48 | PULSE/EDGE  |
| Memory DMA Stream 3 Source<br>Channel                       | 49 | PULSE/EDGE  |
| Memory DMA Stream 3<br>Destination Channel                  | 50 | PULSE/EDGE  |

**Table 8-2:** Masters ADSP-BF60x Trigger List (Continued)

| Description                             | ID | Sensitivity |
|-----------------------------------------|----|-------------|
| EPPI0 Channel 0 DMA                     | 51 | PULSE/EDGE  |
| EPPI0 Channel 1 DMA                     | 52 | PULSE/EDGE  |
| EPPI2 Channel 0 DMA                     | 53 | PULSE/EDGE  |
| EPPI2 Channel 1 DMA                     | 54 | PULSE/EDGE  |
| EPPI1 Channel 0 DMA                     | 55 | PULSE/EDGE  |
| EPPI1 Channel 1 DMA                     | 56 | PULSE/EDGE  |
| PIXC0 Channel 0 DMA                     | 57 | PULSE/EDGE  |
| PIXC0 Channel 1 DMA                     | 58 | PULSE/EDGE  |
| PIXC0 Channel 2 DMA                     | 59 | PULSE/EDGE  |
| PVP0 Camera Pipe Data Out B DMA Channel | 60 | PULSE/EDGE  |
| PVP0 Camera Pipe Data Out C DMA Channel | 61 | PULSE/EDGE  |
| PVP0 Camera Pipe Status Out DMA Channel | 62 | PULSE/EDGE  |
| PVP0 Camera Pipe Control In DMA Channel | 63 | PULSE/EDGE  |
| PVP0 Memory Pipe Data Out DMA Channel   | 64 | PULSE/EDGE  |
| PVP0 Memory Pipe Data In DMA Channel    | 65 | PULSE/EDGE  |
| PVP0 Memory Pipe Status Out DMA Channel | 66 | PULSE/EDGE  |
| PVP0 Memory Pipe Control In DMA Channel | 67 | PULSE/EDGE  |
| PVP0 Camera Pipe Data Out A DMA Channel | 68 | PULSE/EDGE  |

**Table 8-2:** Masters ADSP-BF60x Trigger List (Continued)

| Description                       | ID | Sensitivity |
|-----------------------------------|----|-------------|
| USB0 DMA Status/Transfer Complete | 69 | LEVEL       |
| Reserved                          | 70 |             |
| SEC0 Fault                        | 71 | PULSE/EDGE  |
| Software-driven Trigger 0         | 72 | PULSE/EDGE  |
| Software-driven Trigger 1         | 73 | PULSE/EDGE  |
| Software-driven Trigger 2         | 74 | PULSE/EDGE  |
| Software-driven Trigger 3         | 75 | PULSE/EDGE  |
| Software-driven Trigger 4         | 76 | PULSE/EDGE  |
| Software-driven Trigger 5         | 77 | PULSE/EDGE  |
| PVP0 Status 0                     | 78 | LEVEL       |
| PVP0 Status 1                     | 79 | LEVEL       |
| SWU0 Event                        | 80 | PULSE/EDGE  |
| SWU1 Event                        | 81 | PULSE/EDGE  |
| SWU2 Event                        | 82 | PULSE/EDGE  |
| SWU3 Event                        | 83 | PULSE/EDGE  |
| SWU4 Event                        | 84 | PULSE/EDGE  |
| SWU5 Event                        | 85 | PULSE/EDGE  |
| SWU6 Event                        | 86 | PULSE/EDGE  |
|                                   |    |             |



**Table 8-3:** Slaves ADSP-BF60x Trigger List

| Description              | ID | Sensitivity |
|--------------------------|----|-------------|
| RCU0 System Reset 0      | 0  |             |
| RCU0 System Reset 1      | 1  |             |
| TIMER0 Timer 0           | 2  |             |
| TIMER0 Timer 1           | 3  |             |
| TIMER0 Timer 2           | 4  |             |
| TIMER0 Timer 3           | 5  |             |
| TIMER0 Timer 4           | 6  |             |
| TIMER0 Timer 5           | 7  |             |
| TIMER0 Timer 6           | 8  |             |
| TIMER0 Timer 7           | 9  |             |
| Reserved                 | 10 |             |
| Reserved                 | 11 |             |
| NMI (Core 0) Slave 0     | 12 |             |
| NMI (Core 0) Slave 1     | 13 |             |
| NMI (Core 1) Slave 0     | 14 |             |
| NMI (Core 1) Slave 1     | 15 |             |
| TRU0 Interrupt Request 0 | 16 |             |
| TRU0 Interrupt Request 1 | 17 |             |
| TRU0 Interrupt Request 2 | 18 |             |
| TRU0 Interrupt Request 3 | 19 |             |
| SPORT0 Channel A DMA     | 20 |             |
| SPORT0 Channel B DMA     | 21 |             |

**Table 8-3:** Slaves ADSP-BF60x Trigger List (Continued)

| Description                                        | ID | Sensitivity |
|----------------------------------------------------|----|-------------|
| SPORT1 Channel A DMA                               | 22 |             |
| SPORT1 Channel B DMA                               | 23 |             |
| SPORT2 Channel A DMA                               | 24 |             |
| SPORT2 Channel B DMA                               | 25 |             |
| SPI0 TX DMA Channel                                | 26 |             |
| SPI0 RX DMA Channel                                | 27 |             |
| SPI1 TX DMA Channel                                | 28 |             |
| SPI1 RX DMA Channel                                | 29 |             |
| RSI0 DMA Channel                                   | 30 |             |
| SDU0 DMA                                           | 31 |             |
| Reserved                                           | 32 |             |
| ACM0 Trigger Input 2                               | 33 |             |
| ACM0 Trigger Input 3                               | 34 |             |
| LP0 DMA Channel                                    | 35 |             |
| LP1 DMA Channel                                    | 36 |             |
| LP2 DMA Channel                                    | 37 |             |
| LP3 DMA Channel                                    | 38 |             |
| UART0 Transmit DMA                                 | 39 |             |
| UART0 Receive DMA                                  | 40 |             |
| UART1 Transmit DMA                                 | 41 |             |
| UART1 Receive DMA                                  | 42 |             |
| Memory DMA Stream 0 Source /<br>CRC0 Input Channel | 43 |             |

**Table 8-3:** Slaves ADSP-BF60x Trigger List (Continued)

| Description                                           | ID | Sensitivity |
|-------------------------------------------------------|----|-------------|
| Memory DMA Stream 0 Destination / CRC0 Output Channel | 44 |             |
| Memory DMA Stream 1 Source / CRC1 Input Channel       | 45 |             |
| Memory DMA Stream 1 Destination / CRC1 Output Channel | 46 |             |
| Memory DMA Stream 2 Source Channel                    | 47 |             |
| Memory DMA Stream 2 Destination Channel               | 48 |             |
| Memory DMA Stream 3 Source Channel                    | 49 |             |
| Memory DMA Stream 3 Destination Channel               | 50 |             |
| EPPI0 Channel 0 DMA                                   | 51 |             |
| EPPI0 Channel 1 DMA                                   | 52 |             |
| EPPI2 Channel 0 DMA                                   | 53 |             |
| EPPI2 Channel 1 DMA                                   | 54 |             |
| EPPI1 Channel 0 DMA                                   | 55 |             |
| EPPI1 Channel 1 DMA                                   | 56 |             |
| PIXC0 Channel 0 DMA                                   | 57 |             |
| PIXC0 Channel 1 DMA                                   | 58 |             |
| PIXC0 Channel 2 DMA                                   | 59 |             |
| PVP0 Camera Pipe Data Out B DMA Channel               | 60 |             |

**Table 8-3:** Slaves ADSP-BF60x Trigger List (Continued)

| Description                             | ID | Sensitivity |
|-----------------------------------------|----|-------------|
| PVP0 Camera Pipe Data Out C DMA Channel | 61 |             |
| PVP0 Camera Pipe Status Out DMA Channel | 62 |             |
| PVP0 Camera Pipe Control In DMA Channel | 63 |             |
| PVP0 Memory Pipe Data Out DMA Channel   | 64 |             |
| PVP0 Memory Pipe Data In DMA Channel    | 65 |             |
| PVP0 Memory Pipe Status Out DMA Channel | 66 |             |
| PVP0 Memory Pipe Control In DMA Channel | 67 |             |
| PVP0 Camera Pipe Data Out A DMA Channel | 68 |             |
| SDU0 Slave Trigger                      | 69 |             |
| Reserved                                | 70 |             |
| Core 0 Wakeup Input 0                   | 71 |             |
| Core 0 Wakeup Input 1                   | 72 |             |
| Core 0 Wakeup Input 2                   | 73 |             |
| Core 0 Wakeup Input 3                   | 74 |             |
| Core 1 Wakeup Input 0                   | 75 |             |
| Core 1 Wakeup Input 1                   | 76 |             |
| Core 1 Wakeup Input 2                   | 77 |             |
| Core 1 Wakeup Input 3                   | 78 |             |

**Table 8-3:** Slaves ADSP-BF60x Trigger List (Continued)

| Description | ID | Sensitivity |
|-------------|----|-------------|
| Reserved    | 79 |             |
| SWU0 Event  | 80 |             |
| SWU1 Event  | 81 |             |
| SWU2 Event  | 82 |             |
| SWU3 Event  | 83 |             |
| SWU4 Event  | 84 |             |
| SWU5 Event  | 85 |             |
| SWU6 Event  | 86 |             |
|             |    |             |

## TRU Definitions

### Trigger Master

A trigger master is any system module that provides trigger event indication to the TRU. Trigger events and conditions for assertion are defined by trigger master modules.

### Trigger Master ID

Trigger masters are assigned a unique numeric ID according to their physical connection to the TRU. Trigger master ID 0 is reserved and defined as null.

### Trigger Slave

A trigger slave is any system module that receives a trigger event indication from the TRU. A trigger event response is defined by the trigger slave modules.

## TRU Block Diagram

Trigger masters and the master trigger register (MTR) generate trigger assertions. Each trigger slave has a dedicated slave select register (SSR) that specifies the unique trigger master from which it receives the trigger indication.

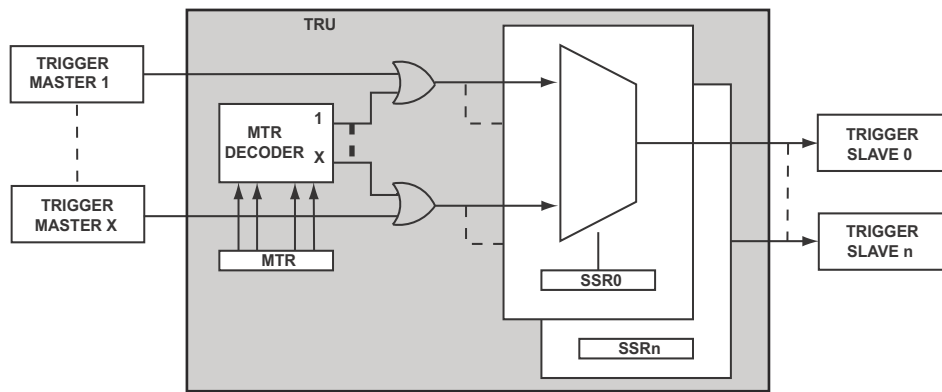


**Figure 8-1:** TRU Block Diagram

## TRU Architectural Concepts

The TRU supports a simple trigger-in/trigger-out model for modules that comply with the triggering functional model. The TRU is the controller of the trigger system. Trigger outputs from trigger masters are mapped to trigger inputs of trigger slaves through a set of programmable registers (SSRn). System modules may be trigger master only, trigger slave only, or trigger master and trigger slave. All of the trigger input and trigger output signals are connected to a Trigger Routing Unit (TRU) which manages the connections of triggers between modules.

## TRU Programming Model

Implementing sequence control using the TRU requires, at a minimum, proper configuration of a trigger slave, a trigger master, and the TRU module itself. The only requirement for the configuration procedure is that the trigger master should be configured and enable as the last step.

The other steps that must be completed are:

- The trigger slave must be configured for response to triggers.
- The TRU must be configured to map the trigger master to the trigger slave through the SSRn registers.
- The trigger master must be configured to generate trigger assertions.
- Alternatively, software triggering may be used for trigger assertion. Software triggers are generated by writing the trigger master ID to the MTR register.

## Programming Concepts

The following concepts will aid in programming the TRU.

- **Trigger Sequence Configuration:** A simple sequence may consist of one Trigger Master and one Trigger Slave. More complex trigger sequences may consist of several Trigger Slaves functioning as Trigger Slave AND Trigger Master. Additionally, trigger sequences may loop back to the original master forming a perpetual sequence.
- **Software Triggering:** Writing a Trigger Master ID to the MTR will generate a trigger within the TRU from the Trigger Master ID specified.
- **Synchronization:** The TRU can be used to synchronize (coarsely) events by mapping multiple Trigger Slaves to the same Trigger Master and/or by generating multiple Trigger Master assertions simultaneously through the MTR.
- **Configuration Protection:** Each SSRn and the GCTL register contain LOCK bits to enable register level write protection when Global Lock is assertion in the SPU.

## Programming Examples

The following example shows the steps to create a trigger.

### Configuring a Simple Trigger Sequence

1. Write to the GCTL register to enable the TRU
2. Write to the SSRn register of a specific trigger slave to assign it to a specific trigger master
3. Enable the trigger slave to wait for and accept a trigger
4. Enable the trigger master to generate a trigger

## TRU Event Control

The TRU is a major part of event control solutions. It is the center of the trigger functional model and may be extended to support the interrupt and fault management models as well.

## TRU Status and Error Signals

The TRU does not have dedicated status and error output signals other than the MMR interface. Slave errors are reported to the master over the standard bus protocol.

# TRU Register Descriptions

Trigger Routing Unit (TRU) contains the following registers.

Table 8-4: TRU Module Registers

| Name        | Description                 |
|-------------|-----------------------------|
| TRU_SSRn    | Slave Select Register       |
| TRU_MTR     | Master Trigger Register     |
| TRU_ERRADDR | Error Address Register      |
| TRU_STAT    | Status Information Register |
| TRU_REVID   | Revision ID Register        |
| TRU_GCTL    | Global Control Register     |

## Slave Select Register

The TRU slave select registers (TRU\_SSRn) each provide slave selection and register locking.

TRU\_SSRn: Slave Select Register - R/W

Reset = 0x0000 0000

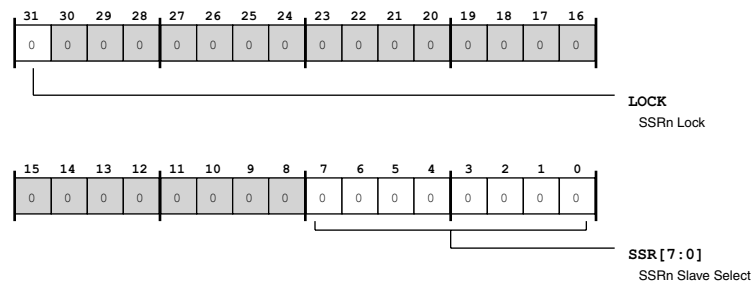


Figure 8-2: TRU\_SSRn Register Diagram

Table 8-5: TRU\_SSRn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | SSRn Lock.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the TRU_SSRn . LOCK bit is enabled, the TRU_SSRn register is read only. |



**Table 8-5:** TRU\_SSRn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | SSR      | SSRn Slave Select.<br>When a TRU_SSRn register is set (for example, TRU_SSRn is set to X), the selected slave n receives trigger X. Trigger X can be from an MTI or can be selected through software by writing X to one of the four fields in the TRU_MTR register. All TRU_MTR and MTI inputs into the TRU are rising edge sensitive. Trigger X must transition from low to high in order for a trigger to be generated. The TRU_SSRn.SSR 8-bit field of each TRU_SSRn selects an input to the trigger multiplexer, producing a TRU output. |

## Master Trigger Register

The TRU master trigger register (TRU\_MTR) permits trigger selection through software by writing X to one of the four fields in the TRU\_MTR register. Trigger inputs to the TRU can be from an MTI or the TRU\_MTR register. These inputs into the TRU are rising edge sensitive. Trigger X must transition from low to high in order for a trigger to be generated. If the global lock is enabled (SPU\_CTL\_GLCK bit =1) and the TRU\_GCTL.LOCK bit is set, the TRU\_MTR register is read only.

### TRU\_MTR: Master Trigger Register - R/W

Reset = 0x0000 0000

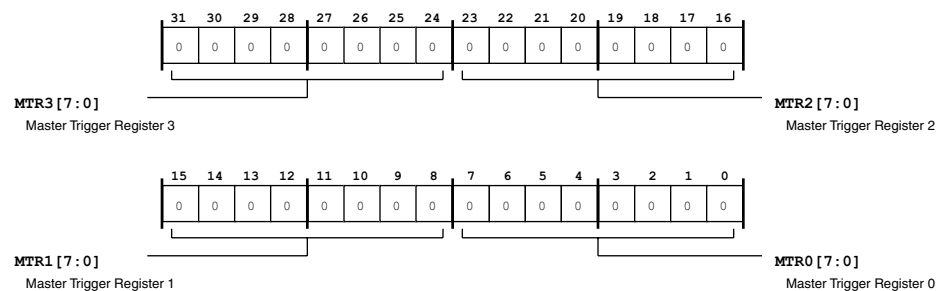


**Figure 8-3:** TRU\_MTR Register Diagram

**Table 8-6:** TRU\_MTR Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:24   | MTR3     | Master Trigger Register 3. |
| 23:16   | MTR2     | Master Trigger Register 2. |
| 15:8    | MTR1     | Master Trigger Register 1. |

**Table 8-6:** TRU\_MTR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 7:0     | MTR0     | Master Trigger Register 0. |

## Error Address Register

The TRU error address register (TRU\_ERRADDR) holds the address from the memory mapped register access generating an access error of TRU registers.

**TRU\_ERRADDR: Error Address Register - R/W**

Reset = 0x0000 0000

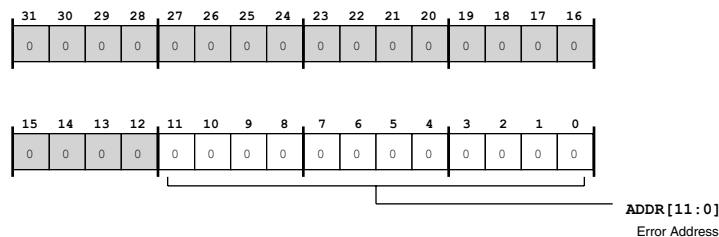


**Figure 8-4:** TRU\_ERRADDR Register Diagram

**Table 8-7:** TRU\_ERRADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:0    | ADDR     | <p>Error Address.</p> <p>The TRU_ERRADDR.ADDR holds the address from the memory mapped register access generating an access error of TRU registers. These errors occur on access to the TRU_SSRn or TRU_MTR registers when these registers are locked. See the TRU_SSRn and TRU_MTR register descriptions for more information about locking. The TRU_ERRADDR register holds the address of the first error to occur. In the event of both TRU_SSRn and TRU_MTR related error types occurring, the TRU_ERRADDR register contains the address of the first error. To re-enable the TRU_ERRADDR register for update, both status bits (TRU_STAT.LWERR and TRU_STAT.ADDRERR) in the TRU_STAT register must be cleared. The TRU_ERRADDR register is writeable, so the address value can be cleared using a standard MMR access.</p> |

## Status Information Register

The TRU status information register (TRU\_STAT) contains the status of TRU\_MTR and TRU\_SSRn register writes and status of bus read/write errors.

**TRU\_STAT: Status Information Register - R/W**

Reset = 0x0000 0000

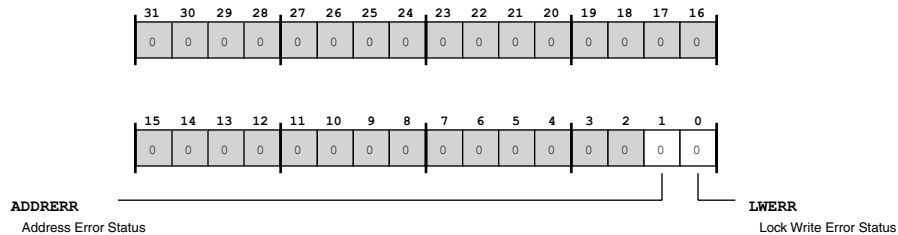


**Figure 8-5:** TRU\_STAT Register Diagram

**Table 8-8:** TRU\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | ADDRERR  | Address Error Status.<br>The TRU_STAT . ADDRERR bit is set when an invalid address is provided for an MMR access while the TRU is selected. Writing a one to this bit clears the error indication. The TRU_ERRADDR register also is updated when an address error occurs during an MMR access while the TRU is selected. |
| 0       | LWERR    | Lock Write Error Status.<br>If TRU_STAT . LWERR is set, a lock write error has occurred. Writing a one to this bit clears the error indication.                                                                                                                                                                          |

## Revision ID Register

The TRU revision ID register (TRU\_REVID) is a read-only register, containing the silicon revision number.

TRU\_REVID: Revision ID Register - R/NW

Reset = 0x0000 0000

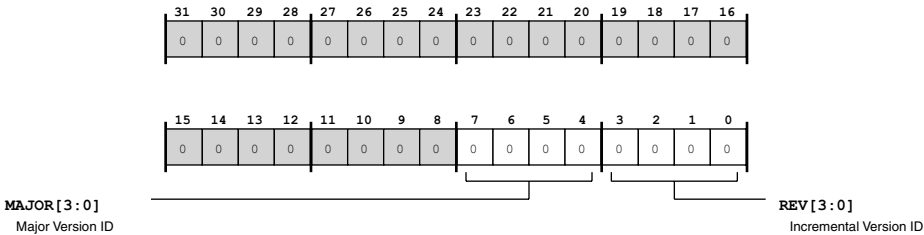


Figure 8-6: TRU\_REVID Register Diagram

Table 8-9: TRU\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:4     | MAJOR    | Major Version ID.       |
| 3:0     | REV      | Incremental Version ID. |

Global Control Register

The TRU global control register (TRU\_GCTL) provides register locking, TRU reset, and TRU enable.

TRU\_GCTL: Global Control Register - R/W

Reset = 0x0000 0000

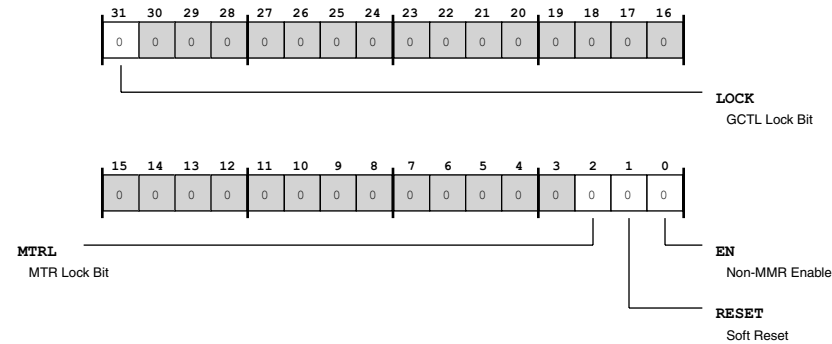


Figure 8-7: TRU\_GCTL Register Diagram

**Table 8-10:** TRU\_GCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | GCTL Lock Bit.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the TRU_GCTL . LOCK bit is enabled, the TRU_GCTL register is read only.                                                                |
| 2       | MTRL     | MTR Lock Bit.<br>If the global lock is enabled (SPU_CTL_GLCK bit =1) and the TRU_GCTL . MTRL bit is enabled, the TRU_MTR register is read only.                                                                  |
| 1       | RESET    | Soft Reset.<br>The TRU_GCTL . RESET bit is write-1-action and triggers a soft reset to all TRU registers.                                                                                                        |
| 0       | EN       | Non-MMR Enable.<br>The TRU_GCTL . EN bit is read/write and must be set for the TRU to propagate trigger events. All TRU register read/write operations continue to operate independent of the TRU_GCTL . EN bit. |



## 9 Static Memory Controller (SMC)

The static memory controller is a protocol converter and data transfer interface between the internal processor bus and the external L3 memory. It provides a glueless interface to a variety of external memories and peripheral devices, including SRAM, ROM, EPROM, NOR flash memory, and FPGA/ASIC devices.

SMC acts as an SCB slave and accesses to SMC are arbitrated by the processor SCB interconnect fabric. On the chip boundary, the SMC is connected to a 25-bit external memory address bus, a 16-bit data bus and memory control signal pins (read, write) including four chip selects via the chip pads.

### SMC Functional Description

Use the following sections to provide functional descriptions of the SMC.

- [SMC Memory Controller Definitions](#)
- [SMC Architectural Concepts](#)

### Features

SMC features include:

- 16-bit I/O width
- Supports up to 64M bytes of SRAM in each of the four external banks
- SMC supports 8-bit data masking writes
- Supports asynchronous access extension (ARDY pin)
- Allows booting from bank 0 (AMS0)
- Supports burst read with programmable burst length of 4, 8 and 16
- Provides flexible timing control through extended timing parameters

### SMC Register List

The static memory controller SMC is a protocol converter and data transfer interface between the internal processor bus and the external L3 memory. The SMC acts as a bus slave and accesses to SMC are arbitrated by the module's system crossbar. On the chip boundary, the SMC is connected to a 25 bit external memory

address bus, a 16-bit data bus and memory control signal pins (read, write) including 4 chip selects. This memory interface can support 64MB of external memory connected to 4 different banks, each bank being controlled by the chip select signal. A set of registers govern SMC operations. For more information on SMC functionality, see the SMC register descriptions.

**Table 9-1:** SMC Module Registers

| Name       | Description                     |
|------------|---------------------------------|
| SMC_GCTL   | Grant Control Register          |
| SMC_GSTAT  | Grant Status Register           |
| SMC_BOCTL  | Bank 0 Control Register         |
| SMC_B0TIM  | Bank 0 Timing Register          |
| SMC_BOETIM | Bank 0 Extended Timing Register |
| SMC_B1CTL  | Bank 1 Control Register         |
| SMC_B1TIM  | Bank 1 Timing Register          |
| SMC_B1ETIM | Bank 1 Extended Timing Register |
| SMC_B2CTL  | Bank 2 Control Register         |
| SMC_B2TIM  | Bank 2 Timing Register          |
| SMC_B2ETIM | Bank 2 Extended Timing Register |
| SMC_B3CTL  | Bank 3 Control Register         |
| SMC_B3TIM  | Bank 3 Timing Register          |
| SMC_B3ETIM | Bank 3 Extended Timing Register |

## SMC Memory Controller Definitions

The SMC contains memory-mapped registers that control the access characteristics for each asynchronous memory bank. The SMC has 1 grant control, 1 read-only grant status register and three memory control registers for each bank. All registers are 32-bit wide. Different Banks can be programmed in different Modes, independently controlled using the functional and cycle time bit settings for each bank.



The SMC\_GCTL register controls the bus grant feature of the controller. There is only one programmable bit in this register which if set disables the Bus grant feature of SMC. It is cleared at reset which means that Bus Grant is enabled by default.

The SMC\_GSTAT indicates the status of the  $\overline{BG}$ ,  $\overline{BR}$ , and  $\overline{BGH}$  pins. There are three bits in this register to reflect the status of these pins.

The SMC provides a separate set of registers SMC\_BxCTL, SMC\_BxTIM and SMC\_BxETIM to control the mode and timing characteristic of each bank independently. SMC\_BxCTL register contains bits for enabling the bank, bits for selecting mode of operation, bits to configure use of ARDY feature, bits for flash page size, synchronous burst and bits for selecting NOR clock frequency.

SMC\_BxCTL also contains bits to control the type of bank select control signal. External FIFO devices often do not have a separate chip select pin. As a result, for a read, the FIFO's output enable ( $\overline{OE}$ ) pin must be connected to the OR (negative AND) of the  $\overline{AMS}$  and the  $\overline{ARE}$ . Similarly, the write case requires an OR between  $\overline{AMS}$  and  $\overline{AWE}$ . The SMC provides this function so that an external OR gate is not required. The appropriate AMS function can be selected for each memory bank region using the BxSELCTRL bits in SMC\_BxCTL register.

SMC\_BxTIM register contains bits to program the setup time, hold time and access time for read and write access to each bank separately. SMC now allows for totally different setup/hold/access times for Reads and Writes. SMC\_BxTIM register allows to control the timing characteristics of the asynchronous memory interface using following parameters.

- Read Setup time – The time between the beginning of a memory cycle ( $\overline{AMSx}$  low) and the read-enable assertion ( $\overline{ARE}$  low).
- Read Hold time – The time between read-enable deassertion ( $\overline{ARE}$  high) and the end of the memory cycle ( $\overline{AMSx}$  high).
- Read Access – The time between read-enable assertion ( $\overline{ARE}$  low) and deassertion ( $\overline{ARE}$  high).
- Write Setup time – The time between the beginning of a memory cycle ( $\overline{AMSx}$  low) and the write-enable assertion ( $\overline{AWE}$  low).
- Write Hold time – The time between write-enable deassertion ( $\overline{AWE}$  high) and the end of the memory cycle ( $\overline{AMSx}$  high).
- Write Access – The time between write-enable assertion ( $\overline{AWE}$  low) and deassertion ( $\overline{AWE}$  high).

Each of these parameters can be programmed in terms of SCLK0 clock cycles.

SMC provides another register for defining additional timing characteristics of control signals by programming the extended timing register SMC\_BxETIM. The SMC\_BxETIM register provides bits to program following timing characteristics.

- Pre-Setup time – The number of cycles  $\overline{AMS}$  is asserted before  $\overline{AOE}$  is asserted.
- Pre-Access time – The number of cycles inserted after  $\overline{ADV}$  is de-asserted, before  $\overline{ARE}$  is asserted for the next access.

- Memory Idle time – The number of bus idle cycles between  $\overline{\text{AMS}}$  de-asserting edge and next asserting edge.
- Memory Transition time – The number of bus idle cycles extending the Idle time cycles in case of the subsequent access has a different data direction or is to different bank.

## SMC Architectural Concepts

Like earlier Blackfin processors the ADSP-BF60x does not directly support 8-bit accesses to the external memories so for this reason the A0 pin has not been provided out of the processor. Hence address lines coming out of the chip are from A1 to A25. But 8-bit accesses are indirectly supported through additional byte enable signals ABE[1:0]. Some 16 bit memory systems allow the Blackfin processor to do 8-bit reads and writes. The byte to be read or written is selected through the ABE0 and ABE1 signals.

The ABE[1:0] pins are both low during all asynchronous reads and 16-bit asynchronous writes. When an asynchronous write is made to the upper byte of a 16-bit memory, ABE1 = 0 and ABE0 = 1. When an asynchronous write is made to the lower byte of a 16-bit memory, ABE1 = 1 and ABE0 = 0.

The SMC can support 64 MB of external memory connected to four different banks—each bank is controlled by the chip select signal. Accesses to the address range of a particular bank cause the chip select of that bank to get asserted.

Table1. Asynchronous Memory Bank Address Range

| Memory Bank Select         | Memory Bank          | Address Start | Address End |
|----------------------------|----------------------|---------------|-------------|
| $\overline{\text{AMS}}[3]$ | Static Memory Bank-3 | 0xBC00 0000   | 0xBFFF FFFF |
| $\overline{\text{AMS}}[2]$ | Static Memory Bank-2 | 0xB800 0000   | 0xBBFF FFFF |
| $\overline{\text{AMS}}[1]$ | Static Memory Bank-1 | 0xB400 0000   | 0xB7FF FFFF |
| $\overline{\text{AMS}}[0]$ | Static Memory Bank-0 | 0xB000 0000   | 0xB3FF FFFF |

Address range allocated to each bank is fixed at 64MByte. However, not all of an enabled memory bank need be populated.

## Avoiding Bus Contention

When using the three-stateable data bus in a system with multiple devices, use care to avoid bus contention. Contention causes excessive power dissipation and can lead to device failure. Contention occurs during the time one device is getting off the bus and another is getting on. If the first device is slow to three-state and the second device is quick to drive, the devices contend. There are two cases where contention can occur. The first case is a read followed by a write to the same memory space. In this case, the data bus drivers can potentially contend with those of the memory device addressed by the read. The second

case is back-to-back reads from two different memory spaces. In this case, the two memory devices addressed by the two reads could potentially contend at the transition between the two read operations.

To avoid contention, program the turnaround time appropriately in the `SMC_BxETIM` registers. This feature allows software to set the number of clock cycles between these types of accesses on a bank-by-bank basis.

Idle time (`BxIT`) applies to similar back to back access types on the same bank. For actual turnaround scenarios idle time and transition time (`BxTT`) function in an accumulated fashion. The sequence of access types and the field(s) that affect access times are shown below.

- a write followed by write to same bank - `B0IT`
- a read followed by read to same bank - `B0IT`
- a write followed by read to same bank - `B0IT + B0TT`
- a read followed by write to same bank - `B0IT + B0TT`
- any access to a given bank followed by any access to a different bank - `B0IT + B0TT`

The reset value of turnaround transition time is 2 cycles. It is recommended that `BxTT` be programmed to a value either greater than or equal to 2 cycles depending upon memory AC-timing specifications. However, `BxTT` may be programmed to 0 ONLY if:

1. No SMC Bank is programmed in synchronous burst mode AND
2. There are either only read accesses or only write accesses possible to external memory devices for the current device configuration/processor operation scenario.

## ARDY Input Control

Each bank can be programmed to sample the ARDY input after the read or write access timer has counted down or to ignore this input signal. If enabled and disabled at the sample window, ARDY can be used to extend the access time as required.

The ARDY input is treated as an asynchronous input, however it must reach the desired value (either asserted or deasserted) more than two SCLK0 cycle before the completion of access time (scheduled rising edge of  $\overline{AWE}$  or  $\overline{ARE}$ ). This determines whether the access is extended or not, the transaction is extended by the assertion of ARDY. Once ARDY (asserted by memory device), is sampled 'HIGH' the total delay between ARDY going high at the pads and  $\overline{ARE}$  being de-asserted at the pads can be a maximum of 5 SCLK cycles.

Asynchronous SRAM writes are also possible with ARDY signal enabled. In Asynchronous SRAM writes with ARDY, the Write Access is extended beyond programmed write access cycles depending on the ARDY signal state. Once ARDY is sampled ASSERTED, the  $\overline{AWE}$  signal is deasserted after 2 CLKOUT cycles and the write access ends.

The polarity of ARDY is programmable on a per-bank basis. Since ARDY is not sampled until an access is in progress to a bank in which the ARDY enable is asserted, ARDY does not need to be driven by default. When

using flash memory, the WAIT input should be connected to ARDY. To avoid stalls in case of erroneous ARDY behavior, the ARDYABRTEN bit can be set to enable the ARDY Abort Counter. If ARDY abort counter is enabled, then the counter starts counting down as soon as the programmed read/write access cycles expire, and times out generating an error if ARDY is not sampled asserted within 64 cycles. This ensures that processor/test does not hang in case ARDY is not sampled correctly.

## Bus Request and Bus Grant

The SMC can relinquish control of the data and address buses to an external device, using the bus request and bus grant protocol. SMC three states its memory interface to allow an external controller to access either external asynchronous or synchronous memory parts.

When an external controller requires access to the static memory bus, it asserts the bus request ( $\overline{BR}$ ) signal. If no internal SMC request is pending, and if the SMC is in the IDLE state, the SMC grants the external bus request. A bus grant is initiated by the following.

- Three-stating the data and address buses and the memory control signals.
- Asserting the bus grant ( $\overline{BG}$ ) signal.

When the static memory bus is granted to an external controller, the SMC stalls all accesses, including instruction fetches and data reads/writes, that address the SMC memory space. Bus grant status may be checked through examination of the `SMC_GSTAT.BGSTAT` bit. This bit can be used by software to check the bus grant status prior to initiating transactions that would be delayed by the external bus grant. When the SMC is ready to perform an access, but is held off because the bus is granted, it asserts the bus grant hang ( $\overline{BGH}$ ) pin. When the external controller releases ( $\overline{BR}$ ), the SMC de-asserts ( $\overline{BG}$ ) and continues servicing access requests.

## Bank-Off Bus Grant

The SMC bus is granted during system reset and after system reset before any one of the SMC banks is enabled. Since all SMC banks are disabled at reset, including bank 0, the SMC bus is granted by default. This bank-off bus grant state does not depend on the status of the external bus request signal. During this bank-off bus grant state, the  $\overline{BG}$  signal is asserted low and the  $\overline{BGH}$  signal is unchanged since internal requests are not possible when all SMC banks are disabled. The state of `SMC_GCTL.BGDIS` bit has no effect during the bank-off bus grant state.

When an SMC bank is enabled, it transitions from the bank-off bus grant state to the active state, during which bus grant depends on external bus request ( $\overline{BR}$ ), on internal access requests, and on SMC settings (for example the `SMC_GCTL.BGDIS` bit). If after SMC operation all SMC banks are disabled, the SMC transitions to the bank-off bus grant state, during which the SMC bus is granted irrespective of external bus request status.

## Bus Request and Bus Grant Protocol Timing

Fig. shows the bus cycles for the bus request and bus grant protocol. The external device bus request signal ( $\overline{BR}$ ) is latched on the rising edge of  $SCLK$ . If the SMC is in the idle state (no access requests pending), the bus grant ( $\overline{BG}$ ) signal is asserted 3.5 cycles later on the falling edge of  $CLKOUT$ . The SMC transitions to the bus grant state, three-stating all SMC buses ( $ADDR$ ,  $DATA$ ,  $\overline{AMS}$ ).

If an internal access request is issued while the SMC has the bus granted, the bus grant hang ( $\overline{BGH}$ ) signal is asserted 1.5 cycles later on the falling edge of  $CLKOUT$ .

When the ( $\overline{BR}$ ) signal is de-asserted by the external memory controller, signaling that the external device is relinquishing control of the bus, the SMC latches the state of ( $\overline{BR}$ ) on the rising edge of  $SCLK$ , and 3.5 cycles later, de-asserts ( $\overline{BG}$ ) and ( $\overline{BGH}$ ) on the falling edge of  $CLKOUT$ , transitions from the bus grant state into the active state and resumes processing pending access requests.

**NOTE: IMPORTANT:** When asserting  $\overline{BR}$  low the external host must wait for 3.5 more  $SYSCLK$  cycles before sampling  $\overline{BG}$ , where a single  $\overline{BG}$  sample is used to initiate multi-cycle bus use.

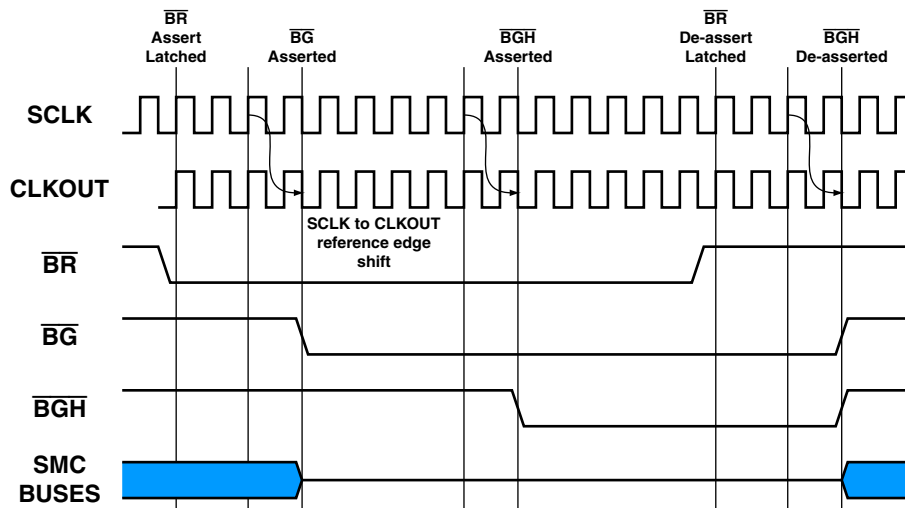


Figure 9-1: SMC Bus Request Bus Grant Protocol

## Disabling Bus Grant to External Memory Controllers

By default, the bus grant feature of the SMC is enabled. To configure the SMC to ignore bus requests and the status of the  $\overline{BR}$  pin, set the  $SMC\_GCTL.BGDIS$  bit.

If the  $SMC\_GCTL.BGDIS$  bit is set while the bus is granted to an external memory controller, the current bus grant status is not affected. The bus continues to be granted until the external memory controller relinquishes control of the bus. At that point, subsequent bus requests are ignored.

## SMC Operating Modes

The SMC supports the following memory devices and operating modes.

- Asynchronous SRAMs
- Flash memories in asynchronous access mode
- Flash memories in synchronous burst mode
- Flash memories in asynchronous page mode
- Asynchronous FIFO devices

### Asynchronous Flash Mode

When the access selected mode is asynchronous flash (MODE = 01), external bank accesses operate exactly the same as in standard asynchronous mode, except for the pin configuration. This mode should be used when accessing burst devices in non-read array modes.

### Synchronous Burst Mode

When synchronous mode access is selected (MODE = 11), synchronous reads are enabled. The burst clock frequency can be configured for SCLK0, SCLK0/2, SCLK0/3 or SCLK0/4. This is the frequency of the clock output and determines the frequency of latching data for subsequent beats of a burst. It does not affect any of the other timing parameters (which are still determined by SMC\_BxTIM and SMC\_BxETIM). During the setup time of an access,  $\overline{ADV}$  is asserted and the burst clock begins running.  $\overline{ADV}$  is asserted for the entire setup time. First rising edge of burst clock (NOR\_CLK) coincides with fall of CLKOUT at the assertion of  $\overline{ADV}$ .

Once the address is latched, the initial burst access occurs based on the read access timing for that bank. The strobe time is then extended by a burst clock duration for each subsequent beat of the burst. Any access in the burst may be extended by connecting the flash WAIT to ARDY. The flash device must be configured to deassert ARDY at the same time that data is valid. There are cases when the NOR\_WAIT is asserted by the flash device in the middle of a read burst (for page crosses). In such cases, Read data latching STOPS until NOR\_WAIT is again deasserted by the flash device, after which data is latched on every rising edge of NOR\_CLK.

The synchronous read may be burst or single mode, depending on the type of transfer requested. Burst access is only supported for back-to-back reads, such as cache line fills (16 words), 64-bit instruction reads (4 words), and DMA reads. Write accesses in synchronous flash burst mode go through as simple asynchronous flash writes. This allows easy programming and re-programming of flash configuration registers while the bank is programmed in synchronous burst mode.

## Asynchronous Page Mode

When asynchronous page mode access is selected (MODE = 10), asynchronous page reads are enabled. Page sizes of 4, 8 and 16 words are supported. When performing a page mode read, the first access in the page proceeds according to the read access time configured in SMC\_BxTIM register. This opens the page, the subsequent reads in that page have a period equal to the page wait states programmed in SMC\_BxETIM register. Besides the start of the setup phase, the read address is incremented at the start of every page cycle.

Page mode access is only supported for back-to-back accesses, such as cache line fills (16 words), 64-bit instruction reads (4 words) and DMA reads. Write accesses in asynchronous page mode are treated as simple asynchronous flash write accesses.

## SMC Event Control

The information for this section is TBD.

## SMC Errors

Accesses to reserved locations and writes to read only registers will result in bus Errors. Bus errors are translated into internal SCB Crossbar errors which in turn get translated into interrupt. To report errors occurring in the slave memory devices (for both this memory interface and the MMR interface as well), the core combines the SCB Crossbar response signals to generate a combined error signal indication which is routed to the fault management unit.

## SMC Programmable Timing Characteristics

This section describes the programmable timing characteristics for the SMC. Timing relationships depend on the programming of the SMC bank registers, whether initiation is from the core or from DMA, and the sequence of transactions (read followed by read, read followed by write, and others).

Note: All memory control, address and data signals are driven out of chip with regard to the falling edge of CLKOUT, except for burst clock. CLKOUT is SCLK0 on the chip pins (pad delayed).

## Asynchronous SRAM Reads and Writes

The figure below shows a basic single write and read operation to an external device with SMC programmed in asynchronous SRAM mode. For the current bank, the programmed time cycles are:

- write setup time = 2 cycles
- write access time = 4 cycles
- write hold time is = 2 cycles

- read setup time = 3 cycles
- read access time = 5 cycles
- read hold time = 1 cycle
- turnaround transition time = 2 cycles
- idle transition time = 0 cycles

The asynchronous SRAM bus cycles proceed as follows.

- At the start of the write setup period, the  $\overline{\text{AMS}}$  signal for the target bank asserts. The write data  $\text{WD0}$ , address  $\text{AW0}$  and byte enables become valid.
- At the end of the setup phase and at the start of the write access period, the write enable  $\overline{\text{AWE}}$  asserts.
- At the end of the programmed write access, the  $\overline{\text{AWE}}$  de-asserts. The target device is assumed to have captured the write data before  $\overline{\text{AWE}}$  de-asserts.
- At the end of the write hold period,  $\overline{\text{AMS}}$  de-asserts since the pending access is a read access, and the turnaround transition time cycles start. The write data and byte enables become invalid within 1 cycle of  $\overline{\text{AMS}}$  de-asserting.
- At the end of turnaround transition time, the read setup period starts with  $\overline{\text{AMS}}$  and  $\overline{\text{AOE}}$  being asserted and a new read address  $\text{AR0}$  being presented on the Address Bus.
- At the start of the read access period, the read enable signal  $\overline{\text{ARE}}$  asserts.
- At the end of the read access period, the  $\overline{\text{ARE}}$  signal de-asserts and the read hold period starts. Read data is latched along with  $\overline{\text{ARE}}$  de-asserting.
- At the end of the read hold period, the chip select  $\overline{\text{AMS}}$  is pulled high and turnaround transition cycles are appended unless there is a pending read request to the same bank.



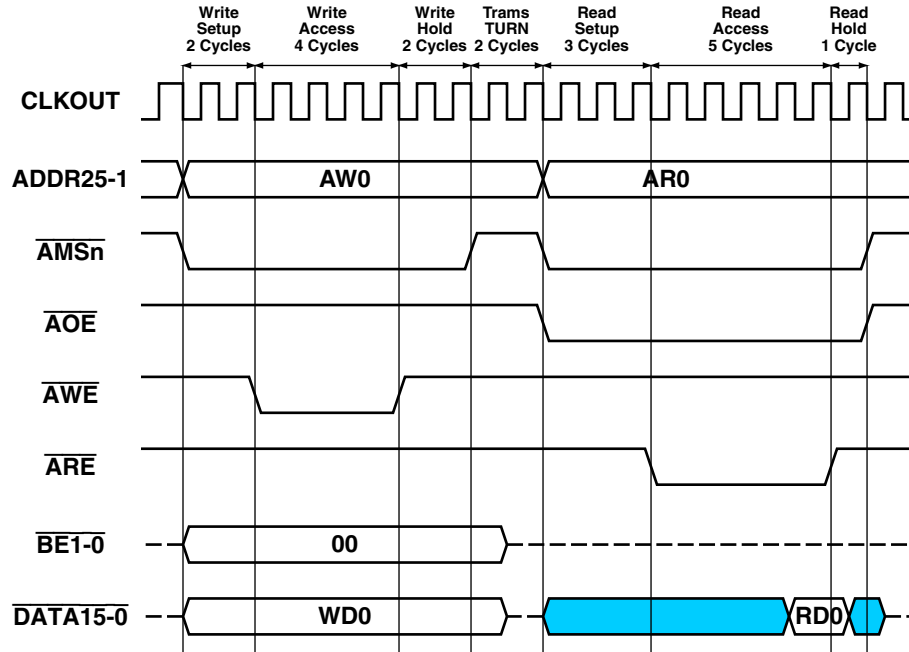


**Figure 9-2:** Basic Asynchronous SRAM Write Followed by Read

## Asynchronous SRAM Reads with IDLE Transition Cycles Inserted

The following figure shows two consecutive asynchronous SRAM mode reads to the same bank separated by programmed IDLE transition time cycles. Programmed cycle times are:

- RST= 2 cycles
- RAT= 4 cycles
- RHT= 1 Cycle
- IDLE trans. time = 2 Cycles

At the start of the IDLE Transition cycle,  $\overline{AMS}$  and  $\overline{AOE}$  are de-asserted. The setup period of the second read starts at the end of the IDLE transition cycle with the assertion of  $\overline{AMS}$  and  $\overline{AOE}$ , and new address on the address bus.

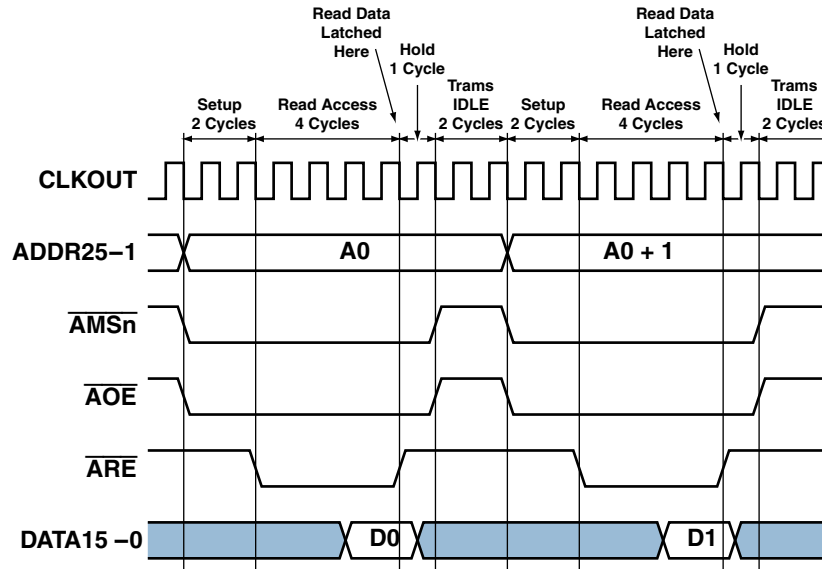


**Figure 9-3:** Asynchronous SRAM Read with IDLE Transition

## High Speed Asynchronous SRAM Read Burst

The following figure shows a high speed asynchronous SRAM read bus cycle. This is typical for SRAM devices with small access times being accessed through SCB read bursts, especially for boot purposes. In this case, the target SMC bank has been programmed with:

- read setup time = 1 cycle
- read access time = 2 cycles
- read hold time = 0
- PREAT= 0
- PREST= 0
- IDLE transition time = 0

The  $\overline{AMS}$  signal asserts at the start of the setup cycle of the first read out of the burst. Since hold time and IDLE transition time have been programmed to 0,  $\overline{AMS}$  does not de-assert until the entire set of reads is over. Only the  $\overline{ARE}$  signal de-asserts periodically for 1 cycle for the setup period. Read address changes to the next address at the start of each individual setup cycle. Read data words are latched at the end of each individual read access period.

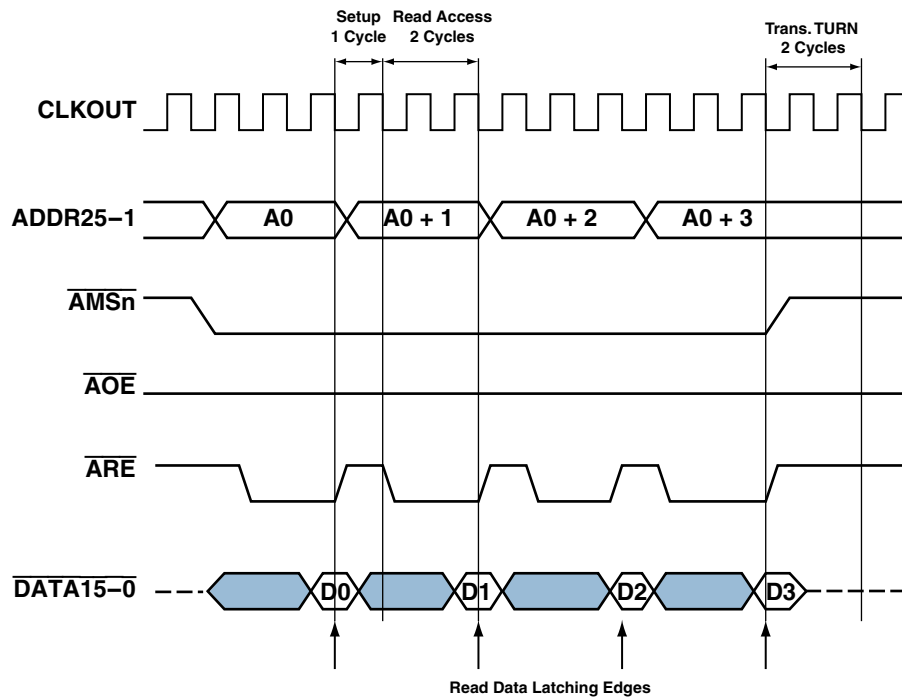


**Figure 9-4:** Fast Asynchronous SRAM Reads, Burst of Four Word

## High Speed Asynchronous SRAM Writes

High speed asynchronous SRAM writes are similar to the high speed read accesses. The bus protocol is shown in the following figure for a write burst of 4 words. Here, write setup time is 1 cycle and write access time has been programmed to 2 cycles. Write hold time, pre access time, pre-setup time and idle transition time have programmed to 0. The chip select signal  $\overline{AMS_n}$  asserts at the start of the entire write burst and de-asserts only at the end of the last individual write access period. Write Address, byte enables and write data for each individual write access are presented onto the bus at the start of each individual write setup cycle. The  $\overline{AWE}$  signal asserts for the write access period and de-asserts during the setup period for each individual data write.

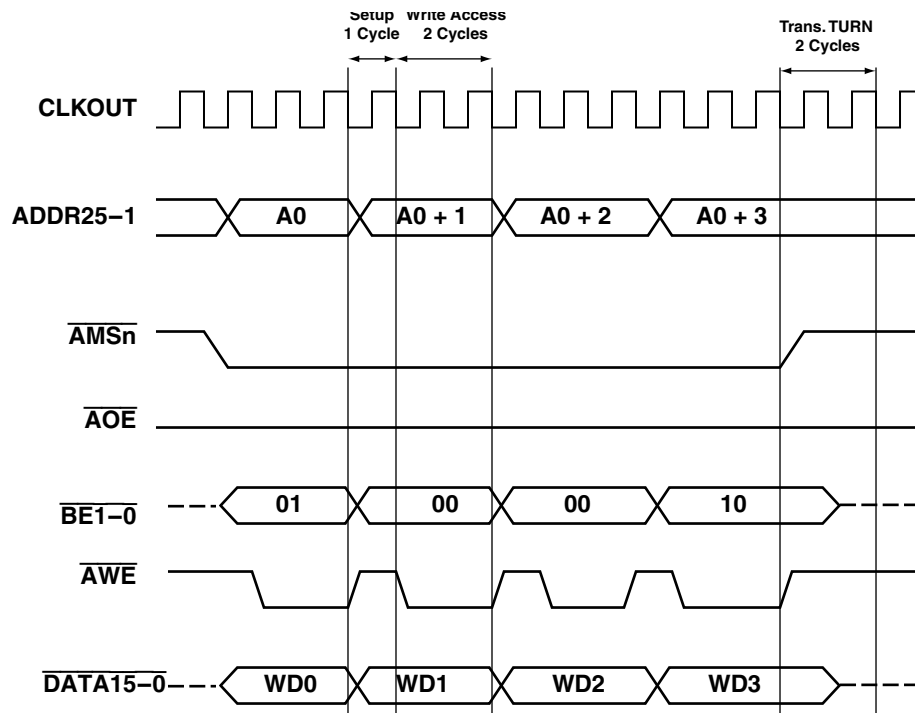


**Figure 9-5:** Fast Asynchronous SRAM Writes

## Asynchronous SRAM Read with ARDY

The following figure shows an extended asynchronous SRAM read bus cycle with ARDY enabled. The programmed SMC bank control parameters are:

- Pre-Setup Time = 1 cycle
- Read Setup Time = 3 Cycles
- Read Access Time = 6 Cycles
- Read Hold Time = 2 Cycles,
- ARDYPOL = 1 (memory is ready when ARDY = 1)

The bus cycles proceed as follows:

- At the start of the pre-setup phase,  $\overline{AMS}$  asserts, and read address A0 is presented on the ADDR bus.
- At the start of the setup period,  $\overline{AOE}$  asserts.
- At the start of the read access,  $\overline{ARE}$  asserts.

- CLKOUT is SCLK which is driven out of the pads. CLKOUT falling edge can be delayed from the internal SCLK fall edge up to 1.5 SCLK cycles for SCLK=125 MHz. All output signals out of the pads for example  $\overline{ARE}$  and  $\overline{AOE}$  are driven with regard to the falling edge of CLKOUT.
- 2 cycles before the programmed number of read access cycles expires, SMC starts sampling ARDY on every rising edge of internal SCLK. Read access is extended ( $\overline{ARE}$  is kept asserted) until ARDY is sampled HIGH.
- Once ARDY (asserted by memory device), is sampled HIGH in SCLK, the read signal is pulled off internally in SCLK domain. The total delay between ARDY going high at the pads and  $\overline{ARE}$  being de-asserted at the pads can be a maximum of 5 SCLK cycles.
- Read data is latched at the falling edge of CLKOUT on the same edge where  $\overline{ARE}$  is deasserted.
- Hold bus cycles start after the  $\overline{ARE}$  is de-asserted.
- At the end of the hold period,  $\overline{AMS}$  and  $\overline{AOE}$  de-assert and the SMC goes into transition state.

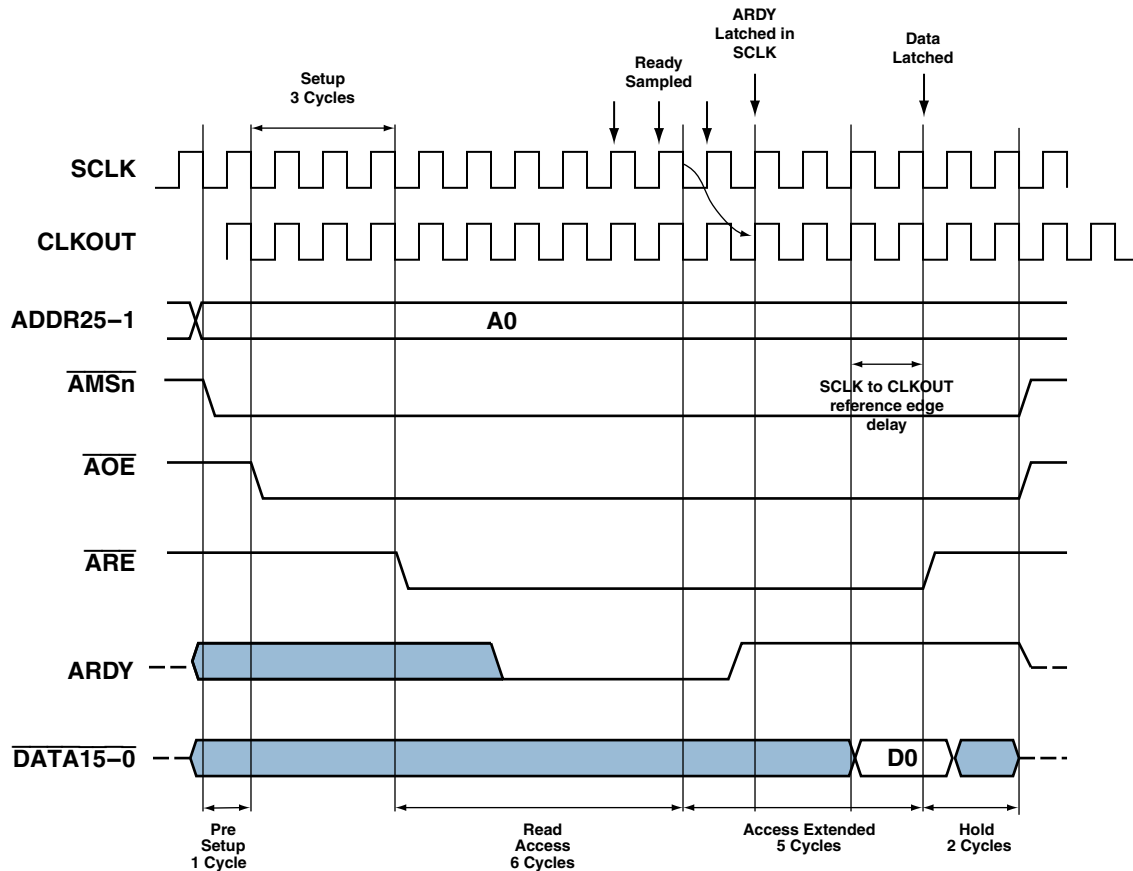


Figure 9-6: Asynchronous SRAM Read with ARDY

## Asynchronous Flash Read Operation

The following figure illustrates a single asynchronous flash mode read bus cycle. In this case, the target SMC bank has been programmed with:

- pre-setup time = 1 cycle
- read setup time = 2 cycles
- pre-access time = 1 cycle
- read access time = 5 cycles
- read hold time = 2 cycles.

The read bus cycle is almost identical to the asynchronous SRAM read bus cycle. The only difference is the behavior of  $\overline{AOE}$  signal which is used as the flash address valid  $\overline{NOR\_ADV}$ .  $\overline{NOR\_ADV}$  asserts at the start of the setup cycle and de-asserts at the end of the setup cycle.

The pre-access cycle inserts 1 cycle gap between the de-assertion of  $\overline{NOR\_ADV}$  and the assertion of Flash Read strobe  $\overline{NOR\_OE}$  at the start of read access. asynchronous flash reads can also be used with  $\overline{ARDY}$  enabled for flash devices which use  $\overline{NOR\_WAIT}$  in asynchronous mode. In that case, the read bus cycle operation is identical to the asynchronous SRAM with  $\overline{ARDY}$  enabled except for the  $\overline{AOE}$  ( $\overline{NOR\_ADV}$ ) signal behavior.

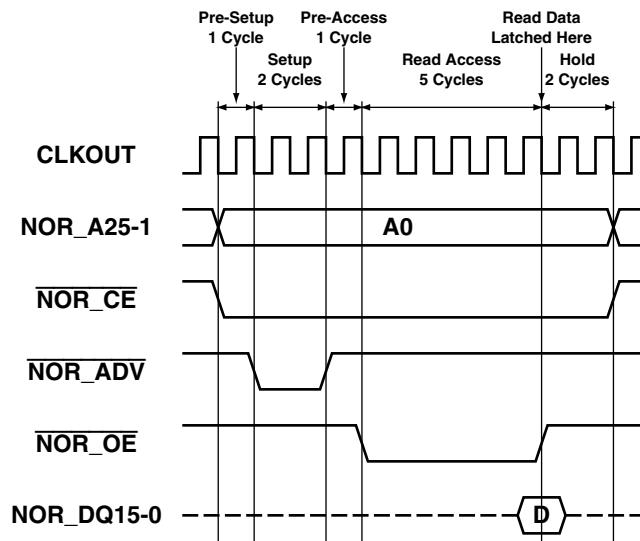


**Figure 9-7:** Asynchronous Flash Read with Pre-Setup and Pre-Access Cycles

The following figure shows a 32-bit read access to a flash device in asynchronous mode. A 32-bit read access to SMC is split into two 16-bit external memory accesses as shown in the figure. For this bank, read setup and read hold are programmed as 2 cycles whereas the read access time is 5 cycles. Note that the flash device chip select  $\overline{NOR\_CE}$  remains asserted for the entire duration of both read accesses, and is de-asserted at the end of the hold period of the second read access.  $\overline{NOR\_ADV}$  is asserted during the setup phase of both read accesses. Read data is latched at the end of the read access period.

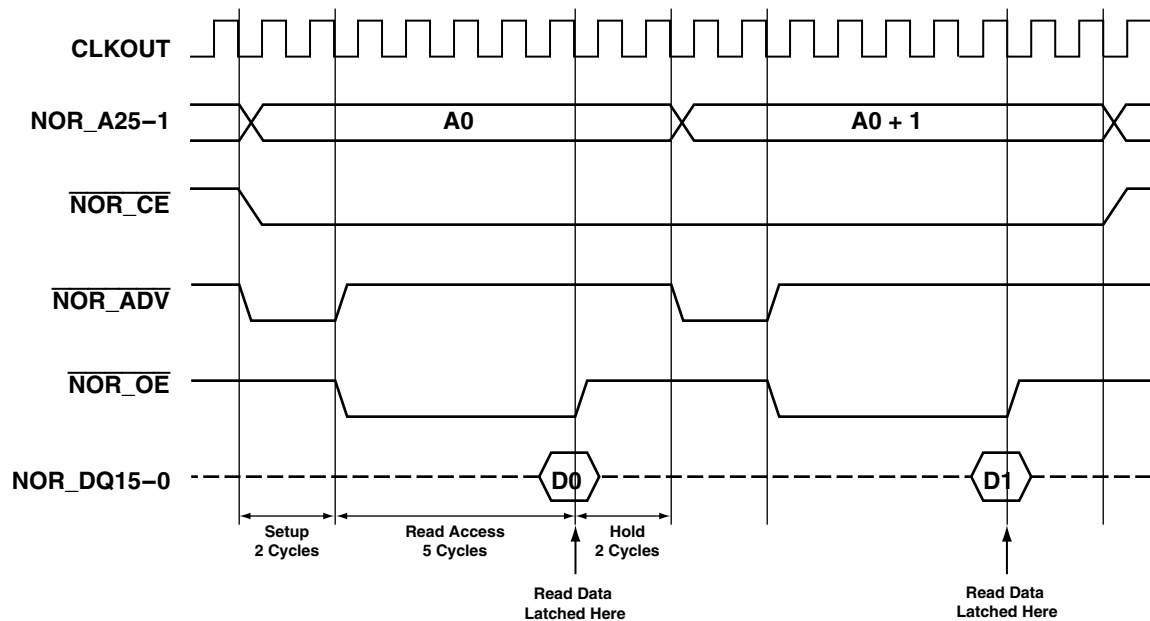


**Figure 9-8:** 32-bit Asynchronous Flash Read

## Asynchronous Flash Write Operation

The following figure shows a single asynchronous flash write bus cycle. For this example, the SMC has been programmed with:

- pre-setup time = 1 cycle
- write setup time = 2 cycles
- write access time = 6 cycles
- write hold time = 2 cycles
- pre-access time = 0

The asynchronous flash write bus cycle is again almost identical to the asynchronous SRAM write. The  $\overline{AWE}$  pin is connected to flash write enable  $\overline{NOR\_WE}$ . However, in asynchronous flash writes, the  $\overline{AOE}$  signal is used as the address valid  $\overline{NOR\_ADV}$  and asserts for the duration of the setup period, unlike in asynchronous SRAM writes where  $\overline{AOE}$  never asserts.

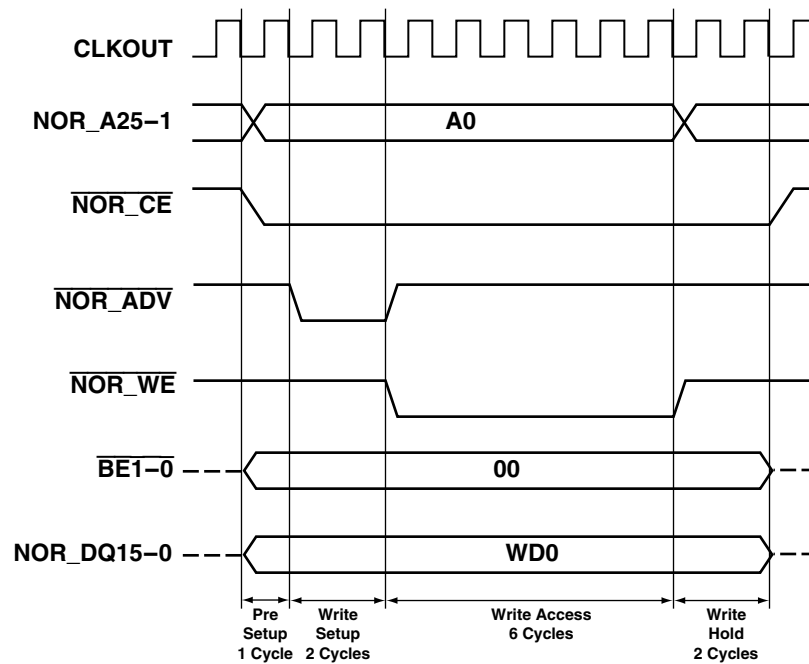


**Figure 9-9:** Asynchronous Flash Write Operation

## Asynchronous Flash Page Mode Read operation

The following figure shows an asynchronous page mode bus read cycle for a burst of 5 reads which are split into 4 reads followed by a single read. The programmed bank parameters are:

- read setup time = 2 cycles
- read access time = 6 cycles
- page wait = 2 cycles
- hold time = 2 cycles

The maximum number of read bursts in a total page access depends on the bank PGSZ bits (00: 4 words, 01: 8 words, 1x: 16 words) The first read access is extended for 3 more page-read cycles whose period is equal to the page wait states. Besides the start of the setup phase, the read address is incremented at the start of every page cycle. Read data is latched with the falling edge of CLKOUT the end of the read access period, and also at the end of the page cycles.



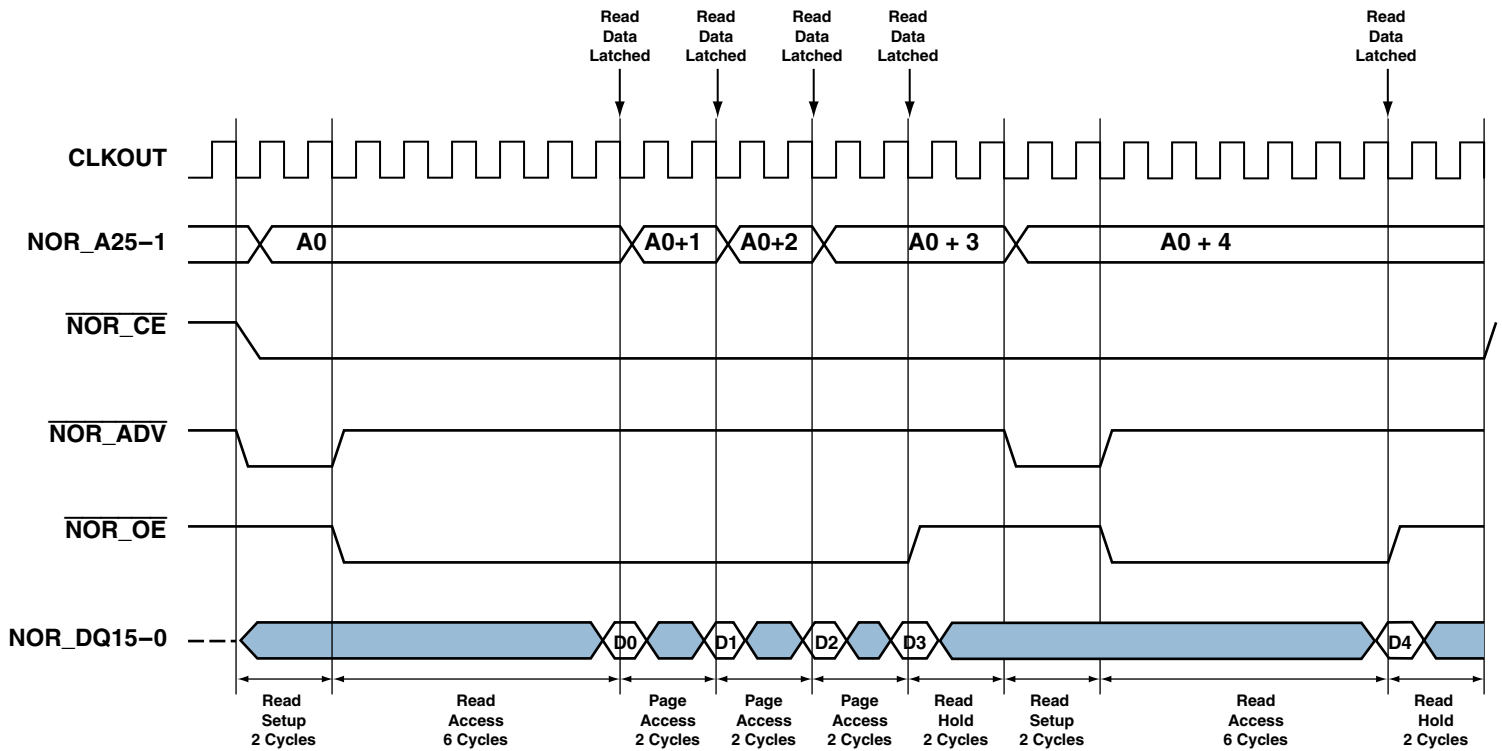


**Figure 9-10:** Asynchronous Page Mode Read Bus Cycle

## Synchronous Burst Mode Read Operation

The following figure shows a synchronous burst mode read operation for a 4 word burst. The Xlatency in the flash device has been programmed to 3, and the burst clock division value is 2. Therefore, the computed read cycle parameters are:

- read setup time = 2 cycles
- read access time = 5 cycles
- read hold time = 1 cycle

The SMC Bus cycles proceed as follows:

- At the start of the read setup period,  $\overline{\text{NOR\_CE}}$  and  $\overline{\text{NOR\_ADV}}$  assert. Read address is presented on the NOR\_A bus. First rising edge of burst clock (NOR\_CLK) coincides with fall of CLKOUT at this edge.
- At the end of the read setup period, which is exactly 1 NOR\_CLK wide,  $\overline{\text{NOR\_ADV}}$  deasserts, and read access starts with  $\overline{\text{NOR\_OE}}$  being asserted.
- Flash de-asserts NOR\_WAIT signal after 2 NOR\_CLK cycles after  $\overline{\text{NOR\_OE}}$  is asserted, as Xlatency has been fixed at 3. (Here, NOR\_WAIT is programmed as active low in flash)

- Read access period ends after 5 CLKOUT cycles, after which NOR\_WAIT is sampled at every rising edge of NOR\_CLK.
- After NOR\_WAIT has been sampled high, read data words are latched on subsequent rising edges of NOR\_CLK, and the read access is extended for 10 CLKOUT cycles, after which the SMC deasserts  $\overline{\text{NOR\_OE}}$  terminating the 4 word burst.
- At the end of the hold period,  $\overline{\text{NOR\_CE}}$  de-asserts, and transition cycles start.
- NOR\_CLK toggle stops with the end of the read access. The last rising edge of NOR\_CLK occurs before  $\overline{\text{NOR\_OE}}$  rises.

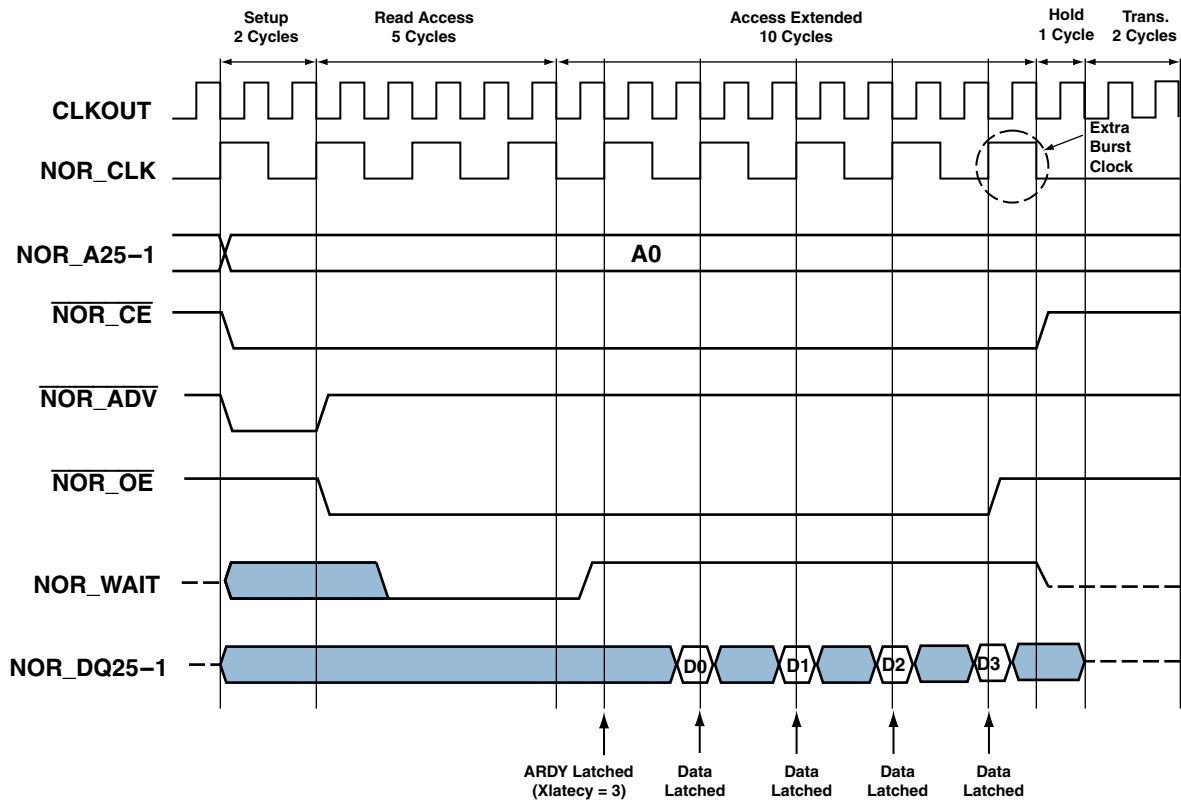


**Figure 9-11:** Synchronous Burst Mode Read Operation, Burst of 4 Words, DIV = 2, XLAT = 3

## Asynchronous FIFO Read and Write Operation

The following figure shows read bus cycles for an asynchronous FIFO device. The SMC bank has been programmed in asynchronous SRAM mode, with  $\text{SELCTRL} = 01$  ( $\overline{\text{AMS}}$  is OR-ed with  $\overline{\text{ARE}}$ ). Other settings are:

- read setup time = 1 cycle
- read access time = 3 cycles
- read hold time = 1 cycle

- idle transition time = 0 cycles
- turnaround transition time = 2 cycles

The  $\overline{AMS}$  signal is connected to the read enable (RE) of the FIFO device, and DATA bus is connected to the output data bus (DQ) of the FIFO.  $\overline{AMS}$  or the FIFO read strobe asserts only for the duration of the read access. Read data is latched at the falling edge of CLKOUT at the end of the read access, when  $\overline{AMS}$  is deasserted.

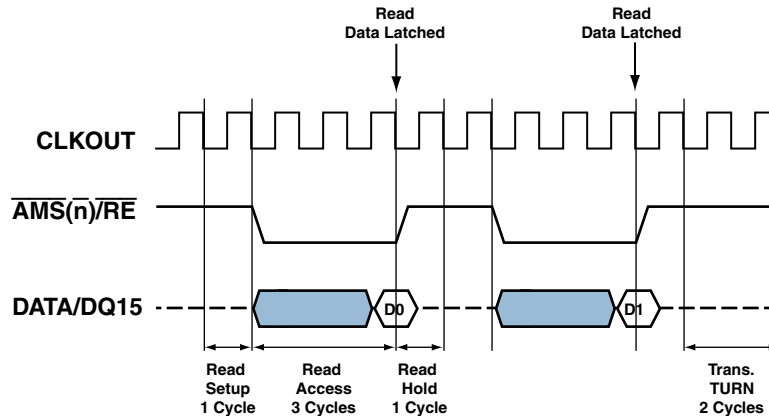


**Figure 9-12:** Asynchronous FIFO Read Bus Cycles

The following figure illustrates write bus cycles for an asynchronous FIFO device. The SMC bank has been programmed in asynchronous SRAM mode, with  $SELCTRL = 10$  ( $\overline{AMS}$  is OR-ed with  $\overline{AW\overline{E}}$ ). Other settings are:

- write setup time = 1 cycle
- write access time = 3 cycles
- write hold time = 1 cycle
- idle transition time = 0
- turnaround transition time = 2 cycles

The  $\overline{AMS}$  signal is connected to the write enable ( $\overline{WE}$ ) of the FIFO device, and DATA bus is connected to the input data bus (DIN) of the FIFO. The  $\overline{AMS}$  signal or the FIFO write strobe asserts only for the duration of the write access. However, write data is asserted at the start of the setup cycle and is taken off at the end of the hold period for each individual write access.

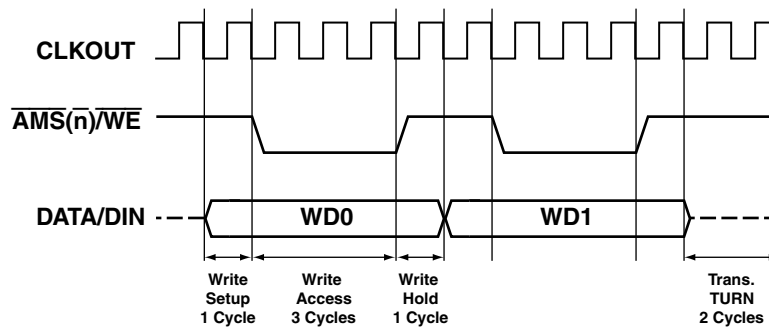


**Figure 9-13:** Asynchronous FIFO Write Bus Cycles

## SMC Programming Model

Following are general guidelines for configuring and enabling the SMC interface. Failure to follow these guidelines can lead to erroneous behavior.

- In asynchronous page mode, the `ARDYEN` bit should always be 0.
- The `ARDY` abort counter should be enabled (the `ARDYBRTEN` bit = 1) whenever `ARDY` is enabled (`ARDYEN` is set to 1). Doing so ensures that the interface does not hang due to erroneous `ARDY` behavior or erroneous sampling of the `ARDY` signal.
- Read access time (`RAT`), write access time (`WAT`), read setup time (`RST`), and write setup time (`WST`) should not be programmed to zero.
- Page mode wait states (`PGWS`) should never be programmed to 0 or 1.
- Program the burst type (`BTYPE`) and the page size bits (`PGSZ`) in the `SMC_BxCTLK` register to match the configurations of the flash device that is being connected to the SMC interface.
- The `RDYPOL` bit should be selected to be the complement of the `WAIT` polarity that is configured in the flash device.
- In asynchronous SRAM and asynchronous flash modes with `ARDY` enabled and where `RHT`, `WHT`, `RAT`, `WAT` are the read and write hold and access times and `IT` and `TT` are the idle and transition times ensure that:
  - $RHT + IT + TT \geq 2$
  - $WHT + IT + TT \geq 2$
  - $RAT \geq 5$
  - $WAT \geq 5$

In synchronous burst mode, program:

- $RST = m \times (BCLKDIV + 1)$ ; where BCLKDIV is the value programmed in the BCLK bit field in SMC\_BxCTL, and  $m = 1, 2, 3, \dots$
- $PREAT = n \times (BCLKDIV + 1)$ ; where BCLKDIV is the value programmed in the BCLK bit field in SMC\_BxCTL, and  $n = 0, 1, 2, \dots$
- $RAT = XLAT \times (BCLKDIV + 1) + 1 - PREAT - RST$ ; where XLAT is the X latency value programmed in the synchronous burst flash device.
- $RHT = BCLKDIV$

Also in synchronous burst mode ensure that:

- $IT + TT \geq 2$
- ARDYEN should always be 1

## SMC Register Descriptions

Static Memory Controller (SMC) contains the following registers.

**Table 9-2:** SMC Module Registers

| Name       | Description                     |
|------------|---------------------------------|
| SMC_GCTL   | Grant Control Register          |
| SMC_GSTAT  | Grant Status Register           |
| SMC_BOCTL  | Bank 0 Control Register         |
| SMC_B0TIM  | Bank 0 Timing Register          |
| SMC_BOETIM | Bank 0 Extended Timing Register |
| SMC_B1CTL  | Bank 1 Control Register         |
| SMC_B1TIM  | Bank 1 Timing Register          |
| SMC_B1ETIM | Bank 1 Extended Timing Register |
| SMC_B2CTL  | Bank 2 Control Register         |
| SMC_B2TIM  | Bank 2 Timing Register          |
| SMC_B2ETIM | Bank 2 Extended Timing Register |

Table 9-2: SMC Module Registers (Continued)

| Name       | Description                     |
|------------|---------------------------------|
| SMC_B3CTL  | Bank 3 Control Register         |
| SMC_B3TIM  | Bank 3 Timing Register          |
| SMC_B3ETIM | Bank 3 Extended Timing Register |

Grant Control Register

The SMC\_GCTL register controls SMC's bus request operation.

SMC\_GCTL: Grant Control Register - R/W

Reset = 0x0000 0000

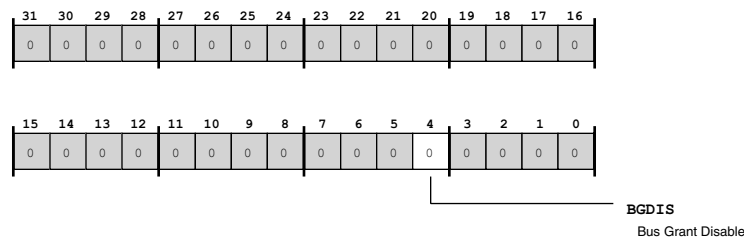


Figure 9-14: SMC\_GCTL Register Diagram

Table 9-3: SMC\_GCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | BGDIS    | Bus Grant Disable.<br>The SMC_GCTL.BGDIS bit disables the SMC's bus request feature, disabling operation of the $\overline{\text{SMC\_BG}}$ , $\overline{\text{SMC\_BR}}$ , and $\overline{\text{SMC\_BGH}}$ pins. |
|         |          | 0                      Enable                                                                                                                                                                                      |
|         |          | 1                      Disable                                                                                                                                                                                     |

Grant Status Register

The SMC\_GSTAT register indicates the status of the  $\overline{\text{SMC\_BG}}$ ,  $\overline{\text{SMC\_BR}}$ , and  $\overline{\text{SMC\_BGH}}$  pins.

**SMC\_GSTAT: Grant Status Register - R/NW**

Reset = 0x0000 0001

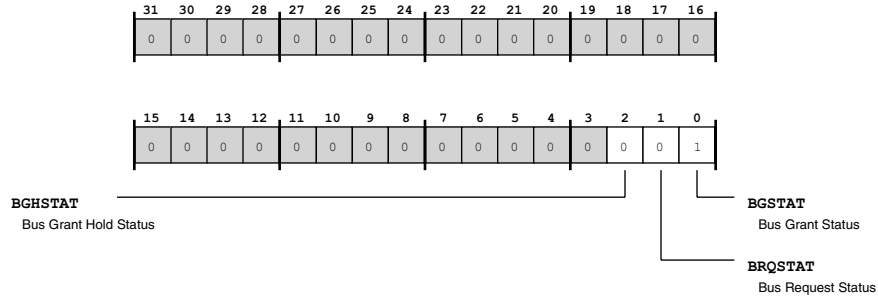


**Figure 9-15:** SMC\_GSTAT Register Diagram

**Table 9-4:** SMC\_GSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | BGHSTAT  | Bus Grant Hold Status.<br>The SMC_GSTAT.BGHSTAT bit indicates the $\overline{\text{SMC\_BGH}}$ pin status. This pin is active when a memory transaction is pending (held) while the bus is granted for another transaction. |
|         |          | 0 Inactive BGH                                                                                                                                                                                                              |
|         |          | 1 Active BGH (transaction pending)                                                                                                                                                                                          |
| 1       | BRQSTAT  | Bus Request Status.<br>The SMC_GSTAT.BRQSTAT bit indicates the $\overline{\text{SMC\_BR}}$ pin status. This pin is active when the bus is requested for a transaction.                                                      |
|         |          | 0 Inactive BR                                                                                                                                                                                                               |
|         |          | 1 Active BR (bus requested)                                                                                                                                                                                                 |
| 0       | BGSTAT   | Bus Grant Status.<br>The SMC_GSTAT.BGSTAT bit indicates the $\overline{\text{SMC\_BG}}$ pin status. This pin is active when bus is granted for a transaction.                                                               |
|         |          | 0 Inactive BG                                                                                                                                                                                                               |
|         |          | 1 Active BG (bus granted)                                                                                                                                                                                                   |

Bank 0 Control Register

The SMC\_BOCTL register enables bank 0 accesses and configures the memory access features for this bank.

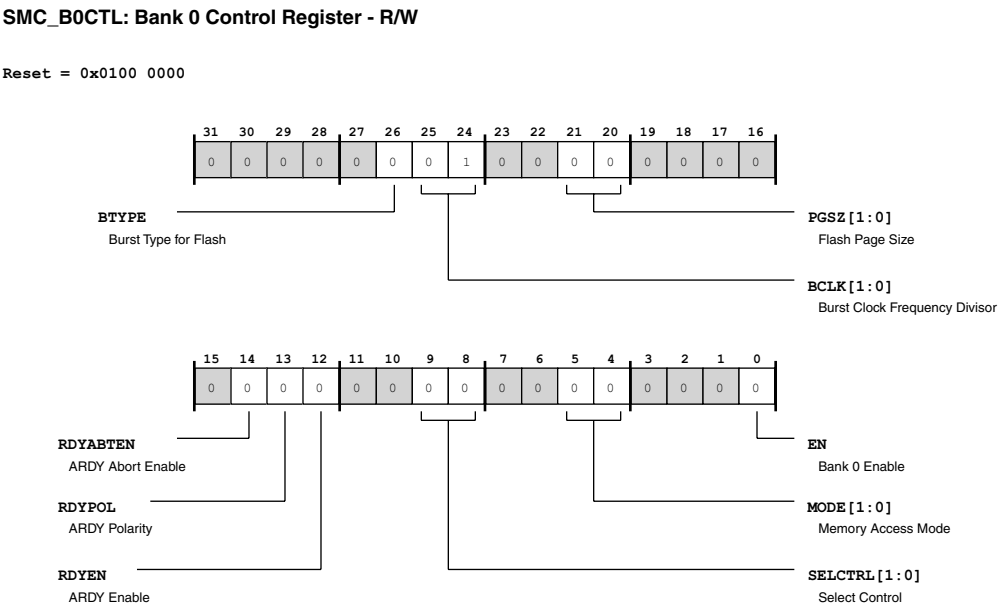


Figure 9-16: SMC\_BOCTL Register Diagram

Table 9-5: SMC\_BOCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 26      | BTYPE    | Burst Type for Flash.<br>The SMC_BOCTL.BTYPE bit selects the burst type that the SMC uses for accesses using sync burst flash protocol. |
|         |          | 0      Wrap                                                                                                                             |
|         |          | 1      Sequential                                                                                                                       |



**Table 9-5:** SMC\_BOCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25:24   | BCLK     | Burst Clock Frequency Divisor.<br>The SMC_BOCTL.BCLK bits select the divisor that the SMC uses to determine the clock frequency for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0 Burst clock = SCLK / 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Burst clock = SCLK / 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2 Burst clock = SCLK / 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 3 Burst clock = SCLK / 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 21:20   | PGSZ     | Flash Page Size.<br>The SMC_BOCTL.PGSZ bits select the flash page size, if page flash or sync burst flash protocol has been enabled (SMC_BOCTL.MODE > 1). Note that the SMC_BOCTL.PGSZ bits must be set to match the flash protocol of the external flash memory device in the system. The typical SMC_BOCTL.PGSZ selection for external devices supporting async flash or async flash page protocols is 4 or 8 words. The typical SMC_BOCTL.PGSZ selection for external devices supporting sync burst flash protocol is 16 words. |
|         |          | 0 4 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 8 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 3 16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 14      | RDYABTEN | ARDY Abort Enable.<br>The SMC_BOCTL.RDYABTEN bit enables the abort counter for the SMC_ARDY pin, if enabled (SMC_BOCTL.RDYEN = 1). After SMC_BOTIM.RAT or SMC_BOTIM.WAT cycles, the SMC starts sampling the SMC_ARDY pin and starts the abort down counter (if enabled). The abort count is 64 cycles of SCLK. If the SMC detects that SMC_ARDY remains de-asserted when the counter expires, the SMC aborts the access and returns an error response back on the system bus.                                                      |
|         |          | 0 Disable abort counter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Enable abort counter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 9-5:** SMC\_BOCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                            |                     |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 13      | RDYPOL   | ARDY Polarity.<br>The SMC_BOCTL.RDYPOL bit selects the polarity (active high or low) for the SMC_ARDY pin, if enabled (SMC_BOCTL.RDYEN =1). When the SMC samples the SMC_ARDY pin in the selective active state, the transaction completes.                                                        |                     |
|         |          | 0                                                                                                                                                                                                                                                                                                  | Low active ARDY     |
|         |          | 1                                                                                                                                                                                                                                                                                                  | High active ARDY    |
| 12      | RDYEN    | ARDY Enable.<br>The SMC_BOCTL.RDYEN bit enables SMC_ARDY pin operation for bank 0 accesses. When enabled, the SMC uses SMC_ARDY (after the access time countdown) to determine completion of access to this memory bank. When disabled, the SMC ignores SMC_ARDY for accesses to this memory bank. |                     |
|         |          | 0                                                                                                                                                                                                                                                                                                  | Disable ARDY        |
|         |          | 1                                                                                                                                                                                                                                                                                                  | Enable ARDY         |
| 9:8     | SELCTRL  | Select Control.<br>The SMC_BOCTL.SELCTRL bits select the handling of the $\overline{\text{SMC\_AMSN}}$ , $\overline{\text{SMC\_ARE}}$ , $\overline{\text{SMC\_AOE}}$ , and $\overline{\text{SMC\_AWE}}$ pins for memory access control.                                                            |                     |
|         |          | 0                                                                                                                                                                                                                                                                                                  | AMS0 only           |
|         |          | 1                                                                                                                                                                                                                                                                                                  | AMS0 or'ed with ARE |
|         |          | 2                                                                                                                                                                                                                                                                                                  | AMS0 or'ed with AOE |
|         |          | 3                                                                                                                                                                                                                                                                                                  | AMS0 or'ed with AWE |

**Table 9-5:** SMC\_B0CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | MODE     | Memory Access Mode.<br>The SMC_B0CTL.MODE bits select the protocol the SMC uses for static memory read/write access. Note that the write protocol for async flash, async flash page, and sync burst flash are all similar; only the read protocols differ for these modes. |
|         |          | 0 Async SRAM protocol                                                                                                                                                                                                                                                      |
|         |          | 1 Async flash protocol                                                                                                                                                                                                                                                     |
|         |          | 2 Async flash page protocol                                                                                                                                                                                                                                                |
|         |          | 3 Sync burst flash protocol                                                                                                                                                                                                                                                |
| 0       | EN       | Bank 0 Enable.<br>The SMC_B0CTL.EN bit enables accesses to the memory in bank 0. When this bit is disabled, accesses to bank 0 return an error response.                                                                                                                   |
|         |          | 0 Disable access                                                                                                                                                                                                                                                           |
|         |          | 1 Enable access                                                                                                                                                                                                                                                            |

## Bank 0 Timing Register

The SMC\_B0TIM register configures bank 0 read and write access, setup, and hold timing for this bank. Note that read and write timing configurations are independent and may differ.

**SMC\_B0TIM: Bank 0 Timing Register - R/W**

Reset = 0x0101 0101

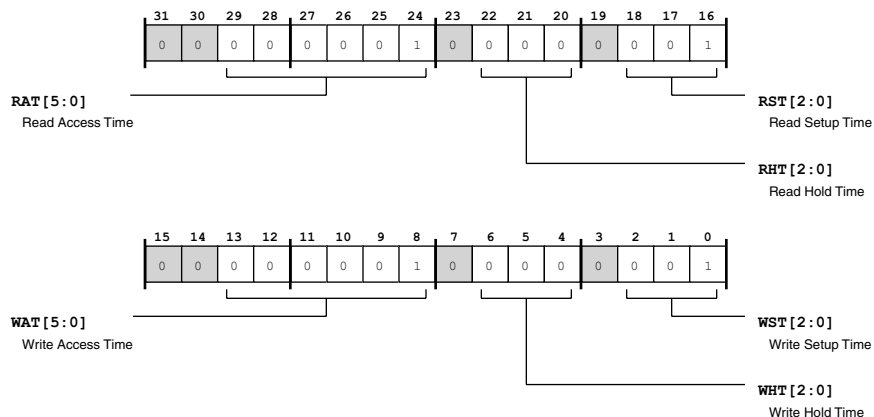


**Figure 9-17:** SMC\_B0TIM Register Diagram

**Table 9-6:** SMC\_B0TIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:24   | RAT      | Read Access Time.<br>The SMC_B0TIM.RAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_ARE}}$ pin for a read access. The access time is from 1 to 63 SCLK cycles.                                                                   |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 63 63 SCLK clock cycles                                                                                                                                                                                                                                                         |
| 22:20   | RHT      | Read Hold Time.<br>The SMC_B0TIM.RHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_ARE}}$ pin before asserting the $\overline{\text{SMC\_AOE}}$ pin for the next access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                           |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                           |

**Table 9-6:** SMC\_B0TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | RST      | Read Setup Time.<br>The SMC_B0TIM.RST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_ARE}}$ pin for an access. The setup time is from 1 to 8 SCLK cycles.                            |
|         |          | 0      8 SCLK clock cycles                                                                                                                                                                                                                                                             |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                             |
| 13:8    | WAT      | Write Access Time.<br>The SMC_B0TIM.WAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AWE}}$ pin for a write access. The access time is from 1 to 63 SCLK cycles.                                                                        |
|         |          | 0      Not supported                                                                                                                                                                                                                                                                   |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 63      63 SCLK clock cycles                                                                                                                                                                                                                                                           |
| 6:4     | WHT      | Write Hold Time.<br>The SMC_B0TIM.WHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AWE}}$ pin before de-asserting the $\overline{\text{SMC\_AOE}}$ pin for the current access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                             |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                             |

**Table 9-6:** SMC\_B0TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2:0     | WST      | Write Setup Time.<br>The SMC_B0TIM.WST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_AWE}}$ pin for a write access. The setup time is from 1 to 8 SCLK cycles. |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                        |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                         |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                        |

Bank 0 Extended Timing Register

The SMC\_B0ETIM register configures extensions to access times and idle times, augmenting the setup, hold, and access times configured with the SMC\_B0TIM register.

SMC\_B0ETIM: Bank 0 Extended Timing Register - R/W

Reset = 0x0002 0200

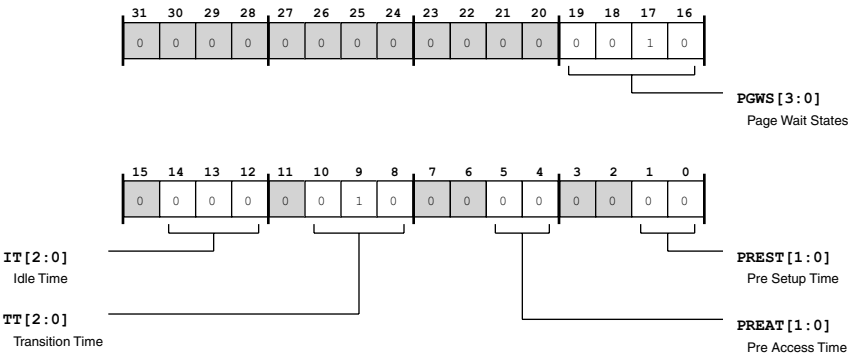


**Figure 9-18:** SMC\_B0ETIM Register Diagram

**Table 9-7:** SMC\_BOETIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:16   | PGWS     | <p>Page Wait States.</p> <p>The SMC_BOETIM.PGWS bits select a page access extension time (in SCLK cycles) that the SMC waits during read accesses when configured for flash page protocol (SMC_BOCTL.MODE =2). The wait time is from 1 to 15 SCLK cycles.</p>                                                                                                                                        |
|         |          | 0      Not supported                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1      1 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 15      15 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                         |
| 14:12   | IT       | <p>Idle Time.</p> <p>The SMC_BOETIM.IT bits select a bus idle time (in SCLK cycles) that the SMC waits between de-asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin and asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin for the next access. Note that the SMC_BOETIM.IT period may be extended using the SMC_BOETIM.TT selection. The idle time is from 0 to 7 SCLK cycles.</p> |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
| 10:8    | TT       | <p>Transition Time.</p> <p>The SMC_BOETIM.TT bits select a bus idle time (in SCLK cycles) that the SMC extends the SMC_BOETIM.IT to allow for the subsequent access either using a different transfer direction or accessing a different bank. The transition time is from 1 to 7 SCLK cycles.</p>                                                                                                   |
|         |          | 0      No bank transition                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |

Table 9-7: SMC\_BOETIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | PREAT    | Pre Access Time.<br>The SMC_BOETIM.PREAT bits select the pre-access time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin before asserting the $\overline{\text{SMC\_ARE/SMC\_AWE}}$ pin for the current access. The pre-access time is from 0 to 3 SCLK cycles. |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |
| 1:0     | PREST    | Pre Setup Time.<br>The SMC_BOETIM.PREST bits select the pre-setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AMSN}}$ pin before asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin for an access. The pre-setup time is from 0 to 3 SCLK cycles.                                      |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |

Bank 1 Control Register

The SMC\_B1CTL register enables bank 1 accesses and configures the memory access features for this bank.

SMC\_B1CTL: Bank 1 Control Register - R/W

Reset = 0x0100 0000

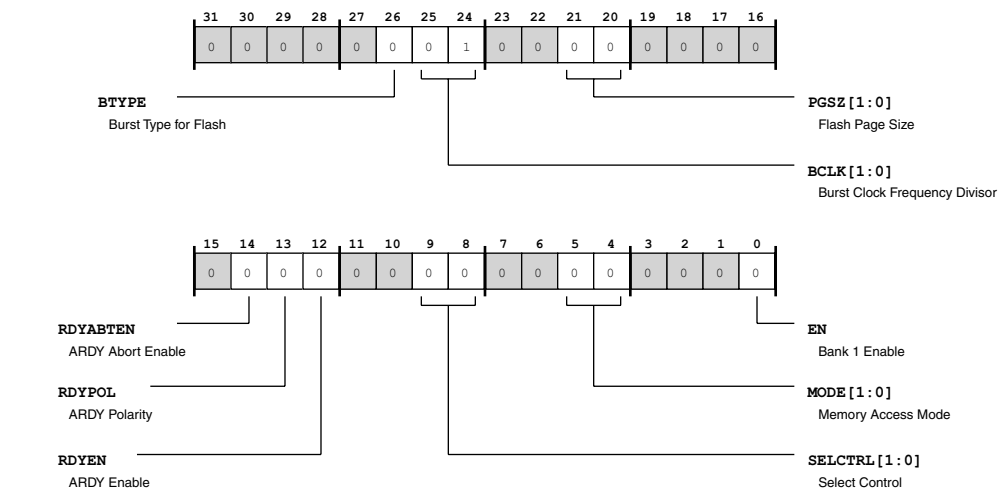


Figure 9-19: SMC\_B1CTL Register Diagram



**Table 9-8:** SMC\_B1CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | BTYP     | Burst Type for Flash.<br>The SMC_B1CTL.BTYP bit selects the burst type that the SMC uses for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0      Wrap                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Sequential                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 25:24   | BCLK     | Burst Clock Frequency Divisor.<br>The SMC_B1CTL.BCLK bits select the divisor that the SMC uses to determine the clock frequency for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0      Burst clock = SCLK / 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Burst clock = SCLK / 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 2      Burst clock = SCLK / 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 3      Burst clock = SCLK / 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21:20   | PGSZ     | Flash Page Size.<br>The SMC_B1CTL.PGSZ bits select the flash page size, if page flash or sync burst flash protocol has been enabled (SMC_B1CTL.MODE > 1). Note that the SMC_B1CTL.PGSZ bits must be set to match the flash protocol of the external flash memory device in the system. The typical SMC_B1CTL.PGSZ selection for external devices supporting async flash or async flash page protocols is 4 or 8 words. The typical SMC_B1CTL.PGSZ selection for external devices supporting sync burst flash protocol is 16 words. |
|         |          | 0      4 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      8 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 2      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 3      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 9-8:** SMC\_B1CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                       |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 14      | RDYABTEN | ARDY Abort Enable.<br>The SMC_B1CTL.RDYABTEN bit enables the abort counter for the SMC_ARDY pin, if enabled (SMC_B1CTL.RDYEN =1). After SMC_B1TIM.RAT or SMC_B1TIM.WAT cycles, the SMC starts sampling the SMC_ARDY pin and starts the abort down counter (if enabled). The abort count is 64 cycles of SCLK. If the SMC detects that SMC_ARDY remains de-asserted when the counter expires, the SMC aborts the access and returns an error response back on the system bus. |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable abort counter |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable abort counter  |
| 13      | RDYPOL   | ARDY Polarity.<br>The SMC_B1CTL.RDYPOL bit selects the polarity (active high or low) for the SMC_ARDY pin, if enabled (SMC_B1CTL.RDYEN =1). When the SMC samples the SMC_ARDY pin in the selective active state, the transaction completes.                                                                                                                                                                                                                                  |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Low active ARDY       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | High active ARDY      |
| 12      | RDYEN    | ARDY Enable.<br>The SMC_B1CTL.RDYEN bit enables SMC_ARDY pin operation for bank 1 accesses. When enabled, the SMC uses SMC_ARDY (after the access time countdown) to determine completion of access to this memory bank. When disabled, the SMC ignores SMC_ARDY for accesses to this memory bank.                                                                                                                                                                           |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable ARDY          |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable ARDY           |

**Table 9-8:** SMC\_B1CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | SELCTRL  | Select Control.<br>The SMC_B1CTL.SELCTRL bits select the handling of the $\overline{\text{SMC\_AMSN}}$ , $\overline{\text{SMC\_ARE}}$ , $\overline{\text{SMC\_AOE}}$ , and $\overline{\text{SMC\_AWE}}$ pins for memory access control.                                    |
|         |          | 0 AMS1 only                                                                                                                                                                                                                                                                |
|         |          | 1 AMS1 or'ed with ARE                                                                                                                                                                                                                                                      |
|         |          | 2 AMS1 or'ed with AOE                                                                                                                                                                                                                                                      |
|         |          | 3 AMS1 or'ed with AWE                                                                                                                                                                                                                                                      |
| 5:4     | MODE     | Memory Access Mode.<br>The SMC_B1CTL.MODE bits select the protocol the SMC uses for static memory read/write access. Note that the write protocol for async flash, async flash page, and sync burst flash are all similar; only the read protocols differ for these modes. |
|         |          | 0 Async SRAM protocol                                                                                                                                                                                                                                                      |
|         |          | 1 Async flash protocol                                                                                                                                                                                                                                                     |
|         |          | 2 Async flash page protocol                                                                                                                                                                                                                                                |
|         |          | 3 Sync burst flash protocol                                                                                                                                                                                                                                                |
| 0       | EN       | Bank 1 Enable.<br>The SMC_B1CTL.EN bit enables accesses to the memory in bank 1. When this bit is disabled, accesses to bank 1 return an error response.                                                                                                                   |
|         |          | 0 Disable access                                                                                                                                                                                                                                                           |
|         |          | 1 Enable access                                                                                                                                                                                                                                                            |

## Bank 1 Timing Register

The SMC\_B1TIM register configures bank 1 read and write access, setup, and hold timing for this bank. Note that read and write timing configurations are independent and may differ.

**SMC\_B1TIM: Bank 1 Timing Register - R/W**

Reset = 0x0101 0101

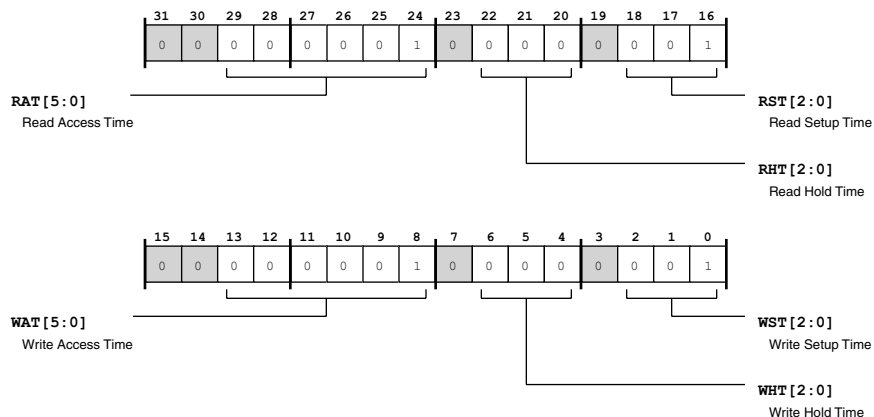


**Figure 9-20:** SMC\_B1TIM Register Diagram

**Table 9-9:** SMC\_B1TIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:24   | RAT      | Read Access Time.<br>The SMC_B1TIM.RAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_ARE}}$ pin for a read access. The access time is from 1 to 63 SCLK cycles.                                                                   |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 63 63 SCLK clock cycles                                                                                                                                                                                                                                                         |
| 22:20   | RHT      | Read Hold Time.<br>The SMC_B1TIM.RHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_ARE}}$ pin before asserting the $\overline{\text{SMC\_AOE}}$ pin for the next access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                           |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                           |

**Table 9-9:** SMC\_BITIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | RST      | Read Setup Time.<br>The SMC_BITIM.RST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_ARE}}$ pin for an access. The setup time is from 1 to 8 SCLK cycles.                            |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                                             |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                                             |
| 13:8    | WAT      | Write Access Time.<br>The SMC_BITIM.WAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AWE}}$ pin for a write access. The access time is from 1 to 63 SCLK cycles.                                                                        |
|         |          | 0                      Not supported                                                                                                                                                                                                                                                   |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 63                     63 SCLK clock cycles                                                                                                                                                                                                                                            |
| 6:4     | WHT      | Write Hold Time.<br>The SMC_BITIM.WHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AWE}}$ pin before de-asserting the $\overline{\text{SMC\_AOE}}$ pin for the current access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                             |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                                             |

Table 9-9: SMC\_B1TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2:0     | WST      | Write Setup Time.<br>The SMC_B1TIM.WST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_AWE}}$ pin for a write access. The setup time is from 1 to 8 SCLK cycles. |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                        |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                         |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                        |

Bank 1 Extended Timing Register

The SMC\_B1ETIM register configures extensions to access times and idle times, augmenting the setup, hold, and access times configured with the SMC\_B1TIM register.

SMC\_B1ETIM: Bank 1 Extended Timing Register - R/W

Reset = 0x0002 0200

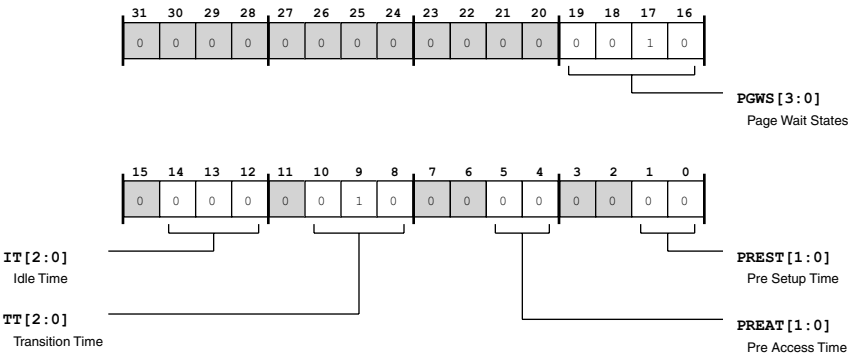


Figure 9-21: SMC\_B1ETIM Register Diagram

**Table 9-10:** SMC\_B1ETIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:16   | PGWS     | <p>Page Wait States.</p> <p>The SMC_B1ETIM.PGWS bits select a page access extension time (in SCLK cycles) that the SMC waits during read accesses when configured for flash page protocol (SMC_B1CTL.MODE =2). The wait time is from 1 to 15 SCLK cycles.</p>                                                                                                                                        |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 1 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 15 15 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                              |
| 14:12   | IT       | <p>Idle Time.</p> <p>The SMC_B1ETIM.IT bits select a bus idle time (in SCLK cycles) that the SMC waits between de-asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin and asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin for the next access. Note that the SMC_B1ETIM.IT period may be extended using the SMC_B1ETIM.TT selection. The idle time is from 0 to 7 SCLK cycles.</p> |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
| 10:8    | TT       | <p>Transition Time.</p> <p>The SMC_B1ETIM.TT bits select a bus idle time (in SCLK cycles) that the SMC extends the SMC_B1ETIM.IT to allow for the subsequent access either using a different transfer direction or accessing a different bank. The transition time is from 1 to 7 SCLK cycles.</p>                                                                                                   |
|         |          | 0 No bank transition                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |

Table 9-10: SMC\_B1ETIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | PREAT    | Pre Access Time.<br>The SMC_B1ETIM.PREAT bits select the pre-access time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin before asserting the $\overline{\text{SMC\_ARE/SMC\_AWE}}$ pin for the current access. The pre-access time is from 0 to 3 SCLK cycles. |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |
| 1:0     | PREST    | Pre Setup Time.<br>The SMC_B1ETIM.PREST bits select the pre-setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AMSN}}$ pin before asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin for an access. The pre-setup time is from 0 to 3 SCLK cycles.                                      |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |

Bank 2 Control Register

The SMC\_B2CTL register enables bank 2 accesses and configures the memory access features for this bank.

SMC\_B2CTL: Bank 2 Control Register - R/W

Reset = 0x0100 0000

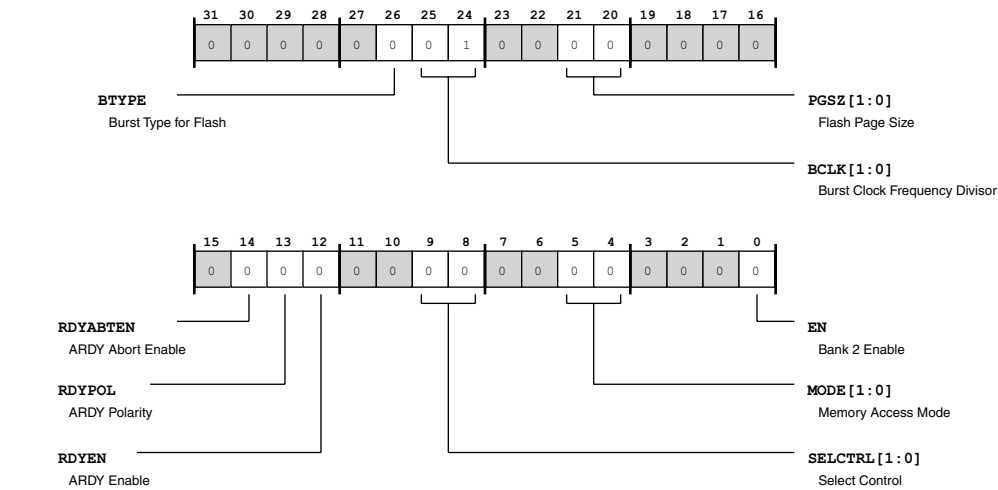


Figure 9-22: SMC\_B2CTL Register Diagram



**Table 9-11:** SMC\_B2CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | BTYP     | Burst Type for Flash.<br>The SMC_B2CTL.BTYP bit selects the burst type that the SMC uses for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0      Wrap                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Sequential                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 25:24   | BCLK     | Burst Clock Frequency Divisor.<br>The SMC_B2CTL.BCLK bits select the divisor that the SMC uses to determine the clock frequency for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0      Burst clock = SCLK / 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Burst clock = SCLK / 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 2      Burst clock = SCLK / 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 3      Burst clock = SCLK / 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21:20   | PGSZ     | Flash Page Size.<br>The SMC_B2CTL.PGSZ bits select the flash page size, if page flash or sync burst flash protocol has been enabled (SMC_B2CTL.MODE > 1). Note that the SMC_B2CTL.PGSZ bits must be set to match the flash protocol of the external flash memory device in the system. The typical SMC_B2CTL.PGSZ selection for external devices supporting async flash or async flash page protocols is 4 or 8 words. The typical SMC_B2CTL.PGSZ selection for external devices supporting sync burst flash protocol is 16 words. |
|         |          | 0      4 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      8 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 2      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 3      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 9-11:** SMC\_B2CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                       |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 14      | RDYABTEN | ARDY Abort Enable.<br>The SMC_B2CTL.RDYABTEN bit enables the abort counter for the SMC_ARDY pin, if enabled (SMC_B2CTL.RDYEN =1). After SMC_B2TIM.RAT or SMC_B2TIM.WAT cycles, the SMC starts sampling the SMC_ARDY pin and starts the abort down counter (if enabled). The abort count is 64 cycles of SCLK. If the SMC detects that SMC_ARDY remains de-asserted when the counter expires, the SMC aborts the access and returns an error response back on the system bus. |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable abort counter |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable abort counter  |
| 13      | RDYPOL   | ARDY Polarity.<br>The SMC_B2CTL.RDYPOL bit selects the polarity (active high or low) for the SMC_ARDY pin, if enabled (SMC_B2CTL.RDYEN =1). When the SMC samples the SMC_ARDY pin in the selective active state, the transaction completes.                                                                                                                                                                                                                                  |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Low active ARDY       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | High active ARDY      |
| 12      | RDYEN    | ARDY Enable.<br>The SMC_B2CTL.RDYEN bit enables SMC_ARDY pin operation for bank 2 accesses. When enabled, the SMC uses SMC_ARDY (after the access time countdown) to determine completion of access to this memory bank. When disabled, the SMC ignores SMC_ARDY for accesses to this memory bank.                                                                                                                                                                           |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable ARDY          |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable ARDY           |

**Table 9-11:** SMC\_B2CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | SELCTRL  | Select Control.<br>The SMC_B2CTL.SELCTRL bits select the handling of the $\overline{\text{SMC\_AMSN}}$ , $\overline{\text{SMC\_ARE}}$ , $\overline{\text{SMC\_AOE}}$ , and $\overline{\text{SMC\_AWE}}$ pins for memory access control.                                    |
|         |          | 0 AMS2 only                                                                                                                                                                                                                                                                |
|         |          | 1 AMS2 or'ed with ARE                                                                                                                                                                                                                                                      |
|         |          | 2 AMS2 or'ed with AOE                                                                                                                                                                                                                                                      |
|         |          | 3 AMS2 or'ed with AWE                                                                                                                                                                                                                                                      |
| 5:4     | MODE     | Memory Access Mode.<br>The SMC_B2CTL.MODE bits select the protocol the SMC uses for static memory read/write access. Note that the write protocol for async flash, async flash page, and sync burst flash are all similar; only the read protocols differ for these modes. |
|         |          | 0 Async SRAM protocol                                                                                                                                                                                                                                                      |
|         |          | 1 Async flash protocol                                                                                                                                                                                                                                                     |
|         |          | 2 Async flash page protocol                                                                                                                                                                                                                                                |
|         |          | 3 Sync burst flash protocol                                                                                                                                                                                                                                                |
| 0       | EN       | Bank 2 Enable.<br>The SMC_B2CTL.EN bit enables accesses to the memory in bank 2. When this bit is disabled, accesses to bank 2 return an error response.                                                                                                                   |
|         |          | 0 Disable access                                                                                                                                                                                                                                                           |
|         |          | 1 Enable access                                                                                                                                                                                                                                                            |

## Bank 2 Timing Register

The SMC\_B2TIM register configures bank 2 read and write access, setup, and hold timing for this bank. Note that read and write timing configurations are independent and may differ.

**SMC\_B2TIM: Bank 2 Timing Register - R/W**

Reset = 0x0101 0101

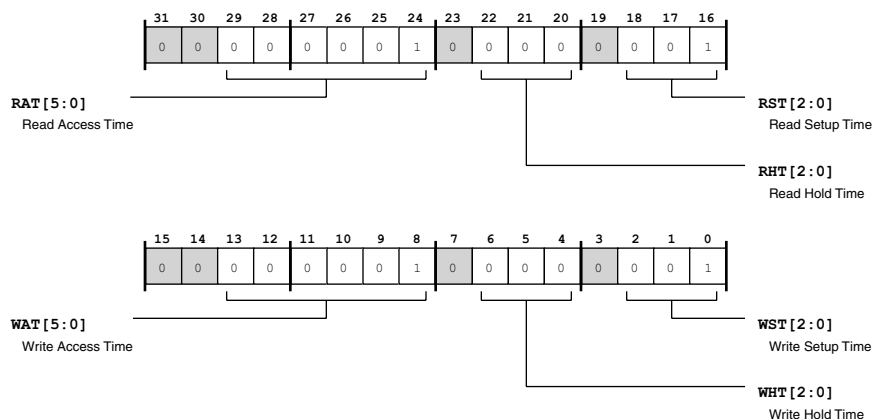


**Figure 9-23:** SMC\_B2TIM Register Diagram

**Table 9-12:** SMC\_B2TIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:24   | RAT      | Read Access Time.<br>The SMC_B2TIM.RAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_ARE}}$ pin for a read access. The access time is from 1 to 63 SCLK cycles.                                                                   |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 63 63 SCLK clock cycles                                                                                                                                                                                                                                                         |
| 22:20   | RHT      | Read Hold Time.<br>The SMC_B2TIM.RHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_ARE}}$ pin before asserting the $\overline{\text{SMC\_AOE}}$ pin for the next access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                           |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                           |

**Table 9-12:** SMC\_B2TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | RST      | Read Setup Time.<br>The SMC_B2TIM.RST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_ARE}}$ pin for an access. The setup time is from 1 to 8 SCLK cycles.                            |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                                             |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                                             |
| 13:8    | WAT      | Write Access Time.<br>The SMC_B2TIM.WAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AWE}}$ pin for a write access. The access time is from 1 to 63 SCLK cycles.                                                                        |
|         |          | 0                      Not supported                                                                                                                                                                                                                                                   |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 63                     63 SCLK clock cycles                                                                                                                                                                                                                                            |
| 6:4     | WHT      | Write Hold Time.<br>The SMC_B2TIM.WHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AWE}}$ pin before de-asserting the $\overline{\text{SMC\_AOE}}$ pin for the current access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                             |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                                              |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                                             |

Table 9-12: SMC\_B2TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2:0     | WST      | Write Setup Time.<br>The SMC_B2TIM.WST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_AWE}}$ pin for a write access. The setup time is from 1 to 8 SCLK cycles. |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                        |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                         |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                        |

Bank 2 Extended Timing Register

The SMC\_B2ETIM register configures extensions to access times and idle times, augmenting the setup, hold, and access times configured with the SMC\_B2TIM register.

SMC\_B2ETIM: Bank 2 Extended Timing Register - R/W

Reset = 0x0002 0200

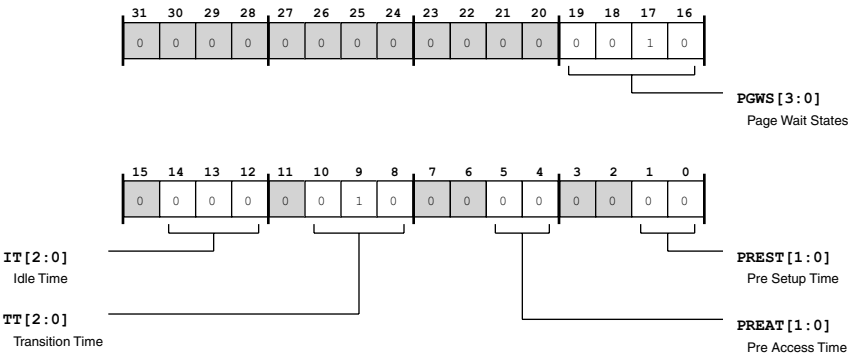


Figure 9-24: SMC\_B2ETIM Register Diagram

**Table 9-13:** SMC\_B2ETIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:16   | PGWS     | <p>Page Wait States.</p> <p>The SMC_B2ETIM.PGWS bits select a page access extension time (in SCLK cycles) that the SMC waits during read accesses when configured for flash page protocol (SMC_B2CTL.MODE =2). The wait time is from 1 to 15 SCLK cycles.</p>                                                                                                                                        |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 1 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 15 15 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                              |
| 14:12   | IT       | <p>Idle Time.</p> <p>The SMC_B2ETIM.IT bits select a bus idle time (in SCLK cycles) that the SMC waits between de-asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin and asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin for the next access. Note that the SMC_B2ETIM.IT period may be extended using the SMC_B2ETIM.TT selection. The idle time is from 0 to 7 SCLK cycles.</p> |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |
| 10:8    | TT       | <p>Transition Time.</p> <p>The SMC_B2ETIM.TT bits select a bus idle time (in SCLK cycles) that the SMC extends the SMC_B2ETIM.IT to allow for the subsequent access either using a different transfer direction or accessing a different bank. The transition time is from 1 to 7 SCLK cycles.</p>                                                                                                   |
|         |          | 0 No bank transition                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                                |

Table 9-13: SMC\_B2ETIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | PREAT    | Pre Access Time.<br>The SMC_B2ETIM.PREAT bits select the pre-access time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin before asserting the $\overline{\text{SMC\_ARE/SMC\_AWE}}$ pin for the current access. The pre-access time is from 0 to 3 SCLK cycles. |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |
| 1:0     | PREST    | Pre Setup Time.<br>The SMC_B2ETIM.PREST bits select the pre-setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AMSN}}$ pin before asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin for an access. The pre-setup time is from 0 to 3 SCLK cycles.                                      |
|         |          | 0                      0 SCLK clock cycles                                                                                                                                                                                                                                                                      |
|         |          | 3                      3 SCLK clock cycles                                                                                                                                                                                                                                                                      |

Bank 3 Control Register

The SMC\_B3CTL register enables bank 3 accesses and configures the memory access features for this bank.

SMC\_B3CTL: Bank 3 Control Register - R/W

Reset = 0x0100 0000

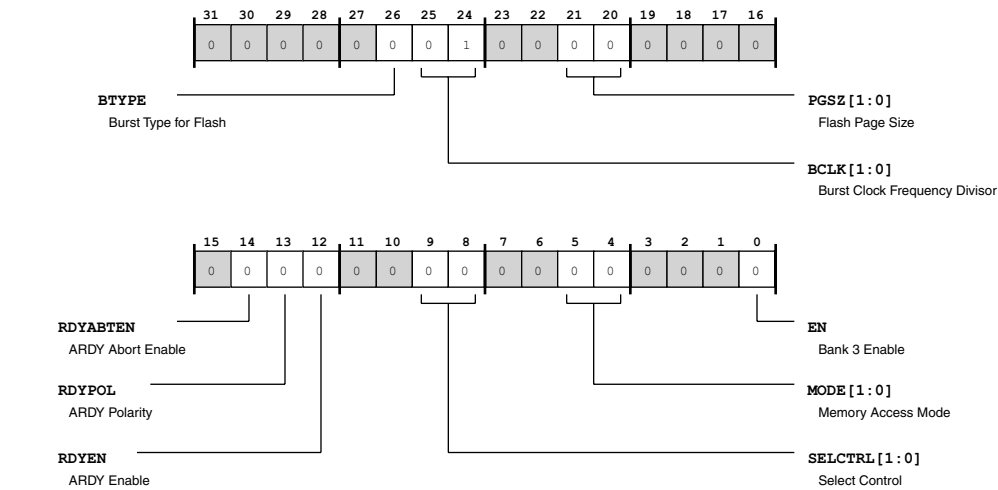


Figure 9-25: SMC\_B3CTL Register Diagram



**Table 9-14:** SMC\_B3CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | BTYP     | Burst Type for Flash.<br>The SMC_B3CTL.BTYP bit selects the burst type that the SMC uses for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0      Wrap                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Sequential                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 25:24   | BCLK     | Burst Clock Frequency Divisor.<br>The SMC_B3CTL.BCLK bits select the divisor that the SMC uses to determine the clock frequency for accesses using sync burst flash protocol.                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0      Burst clock = SCLK / 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Burst clock = SCLK / 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 2      Burst clock = SCLK / 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 3      Burst clock = SCLK / 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21:20   | PGSZ     | Flash Page Size.<br>The SMC_B3CTL.PGSZ bits select the flash page size, if page flash or sync burst flash protocol has been enabled (SMC_B3CTL.MODE > 1). Note that the SMC_B3CTL.PGSZ bits must be set to match the flash protocol of the external flash memory device in the system. The typical SMC_B3CTL.PGSZ selection for external devices supporting async flash or async flash page protocols is 4 or 8 words. The typical SMC_B3CTL.PGSZ selection for external devices supporting sync burst flash protocol is 16 words. |
|         |          | 0      4 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      8 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 2      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 3      16 words                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 9-14:** SMC\_B3CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                       |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 14      | RDYABTEN | ARDY Abort Enable.<br>The SMC_B3CTL.RDYABTEN bit enables the abort counter for the SMC_ARDY pin, if enabled (SMC_B3CTL.RDYEN =1). After SMC_B3TIM.RAT or SMC_B3TIM.WAT cycles, the SMC starts sampling the SMC_ARDY pin and starts the abort down counter (if enabled). The abort count is 64 cycles of SCLK. If the SMC detects that SMC_ARDY remains de-asserted when the counter expires, the SMC aborts the access and returns an error response back on the system bus. |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable abort counter |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable abort counter  |
| 13      | RDYPOL   | ARDY Polarity.<br>The SMC_B3CTL.RDYPOL bit selects the polarity (active high or low) for the SMC_ARDY pin, if enabled (SMC_B3CTL.RDYEN =1). When the SMC samples the SMC_ARDY pin in the selective active state, the transaction completes.                                                                                                                                                                                                                                  |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Low active ARDY       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | High active ARDY      |
| 12      | RDYEN    | ARDY Enable.<br>The SMC_B3CTL.RDYEN bit enables SMC_ARDY pin operation for bank 3 accesses. When enabled, the SMC uses SMC_ARDY (after the access time countdown) to determine completion of access to this memory bank. When disabled, the SMC ignores SMC_ARDY for accesses to this memory bank.                                                                                                                                                                           |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable ARDY          |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable ARDY           |

**Table 9-14:** SMC\_B3CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | SELCTRL  | Select Control.<br>The SMC_B3CTL.SELCTRL bits select the handling of the $\overline{\text{SMC\_AMSN}}$ , $\overline{\text{SMC\_ARE}}$ , $\overline{\text{SMC\_AOE}}$ , and $\overline{\text{SMC\_AWE}}$ pins for memory access control.                                    |
|         |          | 0 AMS3 only                                                                                                                                                                                                                                                                |
|         |          | 1 AMS3 or'ed with ARE                                                                                                                                                                                                                                                      |
|         |          | 2 AMS3 or'ed with AOE                                                                                                                                                                                                                                                      |
|         |          | 3 AMS3 or'ed with AWE                                                                                                                                                                                                                                                      |
| 5:4     | MODE     | Memory Access Mode.<br>The SMC_B3CTL.MODE bits select the protocol the SMC uses for static memory read/write access. Note that the write protocol for async flash, async flash page, and sync burst flash are all similar; only the read protocols differ for these modes. |
|         |          | 0 Async SRAM protocol                                                                                                                                                                                                                                                      |
|         |          | 1 Async flash protocol                                                                                                                                                                                                                                                     |
|         |          | 2 Async flash page protocol                                                                                                                                                                                                                                                |
|         |          | 3 Sync burst flash protocol                                                                                                                                                                                                                                                |
| 0       | EN       | Bank 3 Enable.<br>The SMC_B3CTL.EN bit enables accesses to the memory in bank 3. When this bit is disabled, accesses to bank 3 return an error response.                                                                                                                   |
|         |          | 0 Disable access                                                                                                                                                                                                                                                           |
|         |          | 1 Enable access                                                                                                                                                                                                                                                            |

## Bank 3 Timing Register

The SMC\_B3TIM register configures bank 3 read and write access, setup, and hold timing for this bank. Note that read and write timing configurations are independent and may differ.

**SMC\_B3TIM: Bank 3 Timing Register - R/W**

Reset = 0x0101 0101

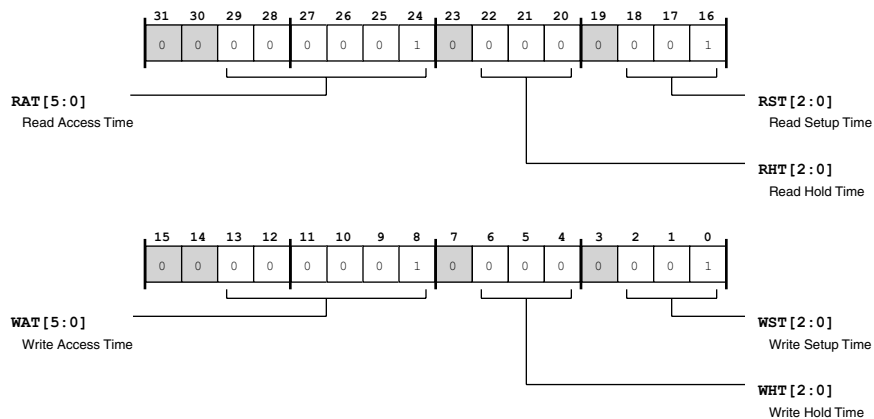


**Figure 9-26:** SMC\_B3TIM Register Diagram

**Table 9-15:** SMC\_B3TIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:24   | RAT      | Read Access Time.<br>The SMC_B3TIM.RAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_ARE}}$ pin for a read access. The access time is from 1 to 63 SCLK cycles.                                                                   |
|         |          | 0 Not supported                                                                                                                                                                                                                                                                 |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 63 63 SCLK clock cycles                                                                                                                                                                                                                                                         |
| 22:20   | RHT      | Read Hold Time.<br>The SMC_B3TIM.RHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_ARE}}$ pin before asserting the $\overline{\text{SMC\_AOE}}$ pin for the next access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0 0 SCLK clock cycles                                                                                                                                                                                                                                                           |
|         |          | 1 1 SCLK clock cycle                                                                                                                                                                                                                                                            |
|         |          | 7 7 SCLK clock cycles                                                                                                                                                                                                                                                           |

**Table 9-15:** SMC\_B3TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | RST      | Read Setup Time.<br>The SMC_B3TIM.RST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_ARE}}$ pin for an access. The setup time is from 1 to 8 SCLK cycles.                            |
|         |          | 0      8 SCLK clock cycles                                                                                                                                                                                                                                                             |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                             |
| 13:8    | WAT      | Write Access Time.<br>The SMC_B3TIM.WAT bits select the access time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AWE}}$ pin for a write access. The access time is from 1 to 63 SCLK cycles.                                                                        |
|         |          | 0      Not supported                                                                                                                                                                                                                                                                   |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 63      63 SCLK clock cycles                                                                                                                                                                                                                                                           |
| 6:4     | WHT      | Write Hold Time.<br>The SMC_B3TIM.WHT bits select the hold time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AWE}}$ pin before de-asserting the $\overline{\text{SMC\_AOE}}$ pin for the current access. The hold time is from 0 to 7 SCLK cycles. |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                             |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                              |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                             |

Table 9-15: SMC\_B3TIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2:0     | WST      | Write Setup Time.<br>The SMC_B3TIM.WST bits select the setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AOE}}$ pin before asserting the $\overline{\text{SMC\_AWE}}$ pin for a write access. The setup time is from 1 to 8 SCLK cycles. |
|         |          | 0                      8 SCLK clock cycles                                                                                                                                                                                                                        |
|         |          | 1                      1 SCLK clock cycle                                                                                                                                                                                                                         |
|         |          | 7                      7 SCLK clock cycles                                                                                                                                                                                                                        |

Bank 3 Extended Timing Register

The SMC\_B3ETIM register configures extensions to access times and idle times, augmenting the setup, hold, and access times configured with the SMC\_B3TIM register.

SMC\_B3ETIM: Bank 3 Extended Timing Register - R/W

Reset = 0x0002 0200

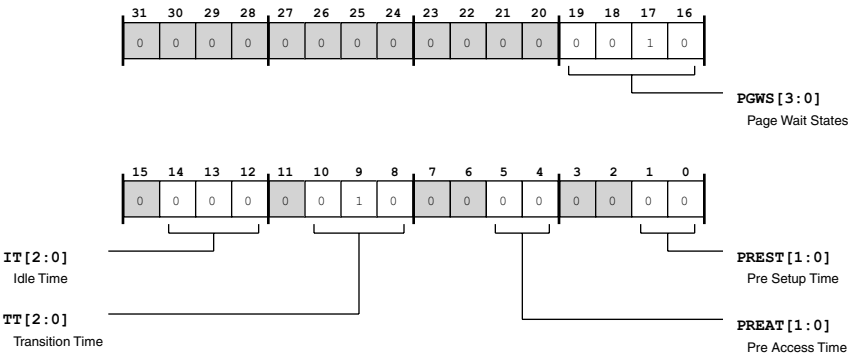


Figure 9-27: SMC\_B3ETIM Register Diagram

**Table 9-16:** SMC\_B3ETIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:16   | PGWS     | <p>Page Wait States.</p> <p>The SMC_B3ETIM.PGWS bits select a page access extension time (in SCLK cycles) that the SMC waits during read accesses when configured for flash page protocol (SMC_B3CTL.MODE =2). The wait time is from 1 to 15 SCLK cycles.</p>                                                                                                                                        |
|         |          | 0      Not supported                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1      1 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 15      15 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                         |
| 14:12   | IT       | <p>Idle Time.</p> <p>The SMC_B3ETIM.IT bits select a bus idle time (in SCLK cycles) that the SMC waits between de-asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin and asserting the <math>\overline{\text{SMC\_AMSN}}</math> pin for the next access. Note that the SMC_B3ETIM.IT period may be extended using the SMC_B3ETIM.TT selection. The idle time is from 0 to 7 SCLK cycles.</p> |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |
| 10:8    | TT       | <p>Transition Time.</p> <p>The SMC_B3ETIM.TT bits select a bus idle time (in SCLK cycles) that the SMC extends the SMC_B3ETIM.IT to allow for the subsequent access either using a different transfer direction or accessing a different bank. The transition time is from 1 to 7 SCLK cycles.</p>                                                                                                   |
|         |          | 0      No bank transition                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      1 SCLK clock cycle                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 7      7 SCLK clock cycles                                                                                                                                                                                                                                                                                                                                                                           |

**Table 9-16:** SMC\_B3ETIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | PREAT    | Pre Access Time.<br>The SMC_B3ETIM.PREAT bits select the pre-access time (in SCLK cycles) that the SMC waits after de-asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin before asserting the $\overline{\text{SMC\_ARE/SMC\_AWE}}$ pin for the current access. The pre-access time is from 0 to 3 SCLK cycles. |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                                                      |
|         |          | 3      3 SCLK clock cycles                                                                                                                                                                                                                                                                                      |
| 1:0     | PREST    | Pre Setup Time.<br>The SMC_B3ETIM.PREST bits select the pre-setup time (in SCLK cycles) that the SMC asserts the $\overline{\text{SMC\_AMSN}}$ pin before asserting the $\overline{\text{SMC\_AOE/ADV}}$ pin for an access. The pre-setup time is from 0 to 3 SCLK cycles.                                      |
|         |          | 0      0 SCLK clock cycles                                                                                                                                                                                                                                                                                      |
|         |          | 3      3 SCLK clock cycles                                                                                                                                                                                                                                                                                      |



# 10 L2 Memory Controller (L2CTL)

The L2 memory controller is a system crossbar slave with multiple SCB ports that access banks of RAM and/or ROM. These interfaces and the memory subsystem run at the L2CLK frequency. The L2 memory controller also has a bus interface to provide access to the L2 controller's control and status registers. Each bank of L2 RAM memory is protected by ECC (error correction and control) logic which ensures that single bit errors can be corrected and multi-bit errors can be detected. For each 32KB bank, a 4Kx14 bits ECC RAM stores the parity bits.

## L2 Memory Controller Functional Description

The L2 memory domain is a unified instruction and data memory and can hold any mixture of code and data required by the system design. The L2 memory domain is accessible by both Blackfin cores through a dedicated 64-bit interface. It operates at half the frequency of the cores. The processor features 256K bytes of L2 SRAM which is ECC-protected and organized in eight banks. Individual banks can be made private to any of the cores or the DMA subsystem. There is also a 32K-byte single-bank ROM in the L2 domain. It contains boot code and safety functions.

## L2CTL Register List

The internal L2 memory controller (L2CTL) includes the controls to manage each L2 memory bank independently. This controller supports ECC and non-ECC operation with support for error tracking (by address or bus ID) and support for error type determination. A set of registers govern L2CTL operations. For more information on L2CTL functionality, see the L2CTL register descriptions.

**Table 10-1:** L2CTL Module Registers

| Name           | Description                    |
|----------------|--------------------------------|
| L2CTL_CTL      | Control Register               |
| L2CTL_ACTL_C0  | Access Control Core 0 Register |
| L2CTL_ACTL_C1  | Access Control Core 1 Register |
| L2CTL_ACTL_SYS | Access Control System Register |
| L2CTL_STAT     | Status Register                |

**Table 10-1:** L2CTL Module Registers (Continued)

| Name           | Description                   |
|----------------|-------------------------------|
| L2CTL_RPCR     | Read Priority Count Register  |
| L2CTL_WPCR     | Write Priority Count Register |
| L2CTL_RFA      | Refresh Address Register      |
| L2CTL_ERRADDR0 | ECC Error Address 0 Register  |
| L2CTL_ERRADDR1 | ECC Error Address 1 Register  |
| L2CTL_ERRADDR2 | ECC Error Address 2 Register  |
| L2CTL_ERRADDR3 | ECC Error Address 3 Register  |
| L2CTL_ERRADDR4 | ECC Error Address 4 Register  |
| L2CTL_ERRADDR5 | ECC Error Address 5 Register  |
| L2CTL_ERRADDR6 | ECC Error Address 6 Register  |
| L2CTL_ERRADDR7 | ECC Error Address 7 Register  |
| L2CTL_ET0      | Error Type 0 Register         |
| L2CTL_EADDR0   | Error Type 0 Address Register |
| L2CTL_ET1      | Error Type 1 Register         |
| L2CTL_EADDR1   | Error Type 1 Address Register |

## ADSP-BF60x L2CTL Interrupt List

**Table 10-2:** ADSP-BF60x L2CTL Interrupt List

| Description      | ID | DMA Channel | Sensitivity |
|------------------|----|-------------|-------------|
| L2CTL0 ECC Error | 4  |             | LEVEL       |

## L2 Memory Controller Block Diagram

The L2 Controller has 2 system crossbar ports. Port 0 is a 64-bit interface that handles core traffic. Port 1 is a 32-bit interface that connects through DMA access. Each port has a read channel and a write channel. Additionally, the read channel has a read address channel and a read data channel. Similarly, the write channel has write address, write data and write response channels.

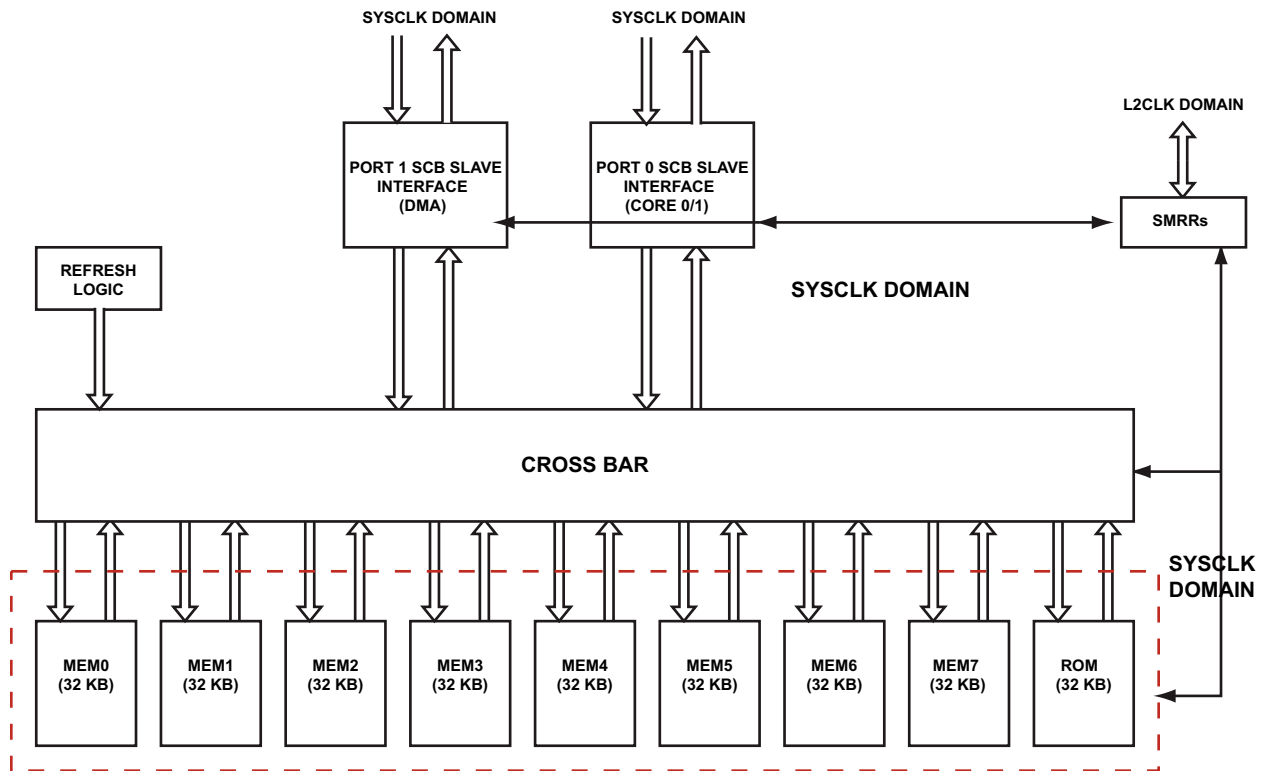


**Figure 10-1:** L2 Memory Controller Block Diagram

RAM is organized in multiple banks, and each bank has 32K Bytes of data. Within each bank, data is organized into 4096 words, with each word comprising 64 bits + 14 bits of ECC. ROM memory does not contain any ECC bits. All access to L2 memories (both reads and writes) is always 64 bits in width. To support 8-, 16- and 32-bit writes, the controller generates a write mask.

## L2 Memory Controller Architectural Concepts

The following sections describe L2 memory controller architecture features, such as access characteristics, access ordering, locked access, and cache support.

### Access Characteristics

The L2 controller interface converts all 8-, 16- and 32-bit accesses to 64-bit accesses. Additionally, 8-, 16- and 32-bit bursts are converted to an equivalent internal 64-bit access. For example, a 64-bit address-aligned burst of 8-bit accesses of burst length 8 are converted to a single 64-bit access.

The L2 controller supports restriction of both core and DMA write accesses to a particular memory bank. The write access by a port to a particular RAM bank can be disabled by setting the appropriate bank write disable bit in that port's L2\_ACTLx register. Illegal access attempts generate an error response.

Memory writes complete in a single L2CLK cycle while reads complete in two cycles.

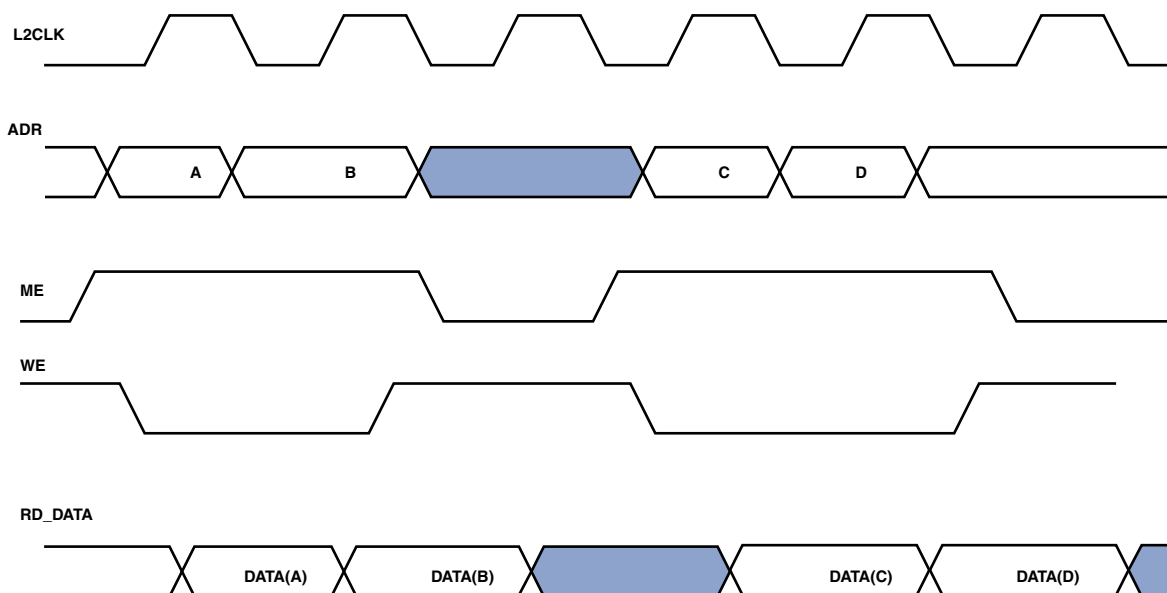


**Figure 10-2: RAM/ROM Read Timing**

## Cache Support

The L2 memory controller does not provide cache support.

## L2 Read/Write Latency and Throughput

The L2 memory design is optimized for burst access at the crossbar interface. Write data of 8/16/32-bit is buffered and converted to equivalent 64-bit access. This conversion creates modulo-32-bit writes if the starting addresses are 32-bit aligned. A single 8- or 16-bit access, or a non-32-bit address-aligned 8-bit or 16-bit burst access to an ECC-enabled bank creates an additional latency of two L2CLK cycles. No extra latency is seen if ECC is disabled.

**NOTE:** Continuous 8/16-bit core access to an ECC-enabled L2 bank is not recommended from a throughput perspective.

If two cores simultaneously try to access L2 for the same kind of access (both read or both write), even to different banks, only one core access is allowed at a time, as there is only 1 read port and 1 write port shared between the cores. However, if one core issues a write and the other issues a read, then access can proceed simultaneously, and inside L2 there is no extra latency, as long as the accesses are to different banks (assuming DMA is also to a non-conflicting bank).

If a core and DMA both access the same bank, then the best access rate that DMA can achieve is one 64-bit access in every three L2CLK cycles during the conflict period. This is achieved by programming the L2CTL\_RPCR.RPC0 and L2CTL\_WPCR.WPC0 bits to 0, while programming the L2CTL\_RPCR.RPC1 and the L2CTL\_WPCR.WPC1 bits to 1.

## Arbitration and Priority

Each bank of L2 RAM/ROM has an arbiter which receives requests from the two crossbar ports. The arbiters for all RAM banks are identical, but the ROM arbiter is simpler in that it only deals with read channels.

The RAM arbiter can receive four requests from the SCB ports (two read channel requests and two write channel requests), as well as a refresh request. Requesting channels are granted depending on their priority. The grants are valid for a particular cycle for writes and address reads, and valid for the next cycle for data reads.

Each arbiter follows a fixed priority scheme for giving grants when more than one channel requests the same bank. The arbiter also supports priority elevation through Urgent Priority requests.

The table below shows the priority for fixed priority mode (with urgent priority disabled) for each SCB channel.

**Table 10-3:** Fixed Priority

| Channel              | Priority Level |
|----------------------|----------------|
| L2 Refresh Request   | 5 (highest)    |
| Port 0 Read Channel  | 4              |
| Port 0 Write Channel | 3              |
| Port 1 Read Channel  | 2              |
| Port 1 Write Channel | 1 (lowest)     |

The arbiters also support priority elevation for a particular channel that has been starved of grants for a very large number of L2CLK cycles. If a channel does not get a grant for “N” cycles after its request, then that channel can elevate the priority of its request by issuing a urgent priority request. This causes that particular channel to become the highest priority master for the next grant cycle (this is pipelined arbitration for urgent priority). The number of cycles “N”, after which the priority is elevated, can be programmed for each channel separately using the L2CTL\_RPCR and L2CTL\_WPCR registers.

If there is an access conflict between the core and DMA to the same memory bank in the fixed priority arbitration scheme, with core activity always prioritized over DMA activity, and with the pipelined implementation of urgent priority, the best grant rate DMA can achieve is 1 in 3 L2CLK cycles during the conflict period. This can be achieved by programming L2CTL\_RPCR.RPCx and L2CCTL\_WPCR.WPCx appropriately.

Urgent priority requests can be disabled by setting L2CTL\_CTL.DISURP bit. This disables the urgent priority requests for all port channels. Each channel can also be prevented from raising the urgent priority request through the priority count register for the specific channel. However, there is no support for disabling urgent priority for a specific memory bank arbiter.

**Table 10-4:** Fixed Priority With Priority Elevation

| Channel                             | Priority Level |
|-------------------------------------|----------------|
| L2 Refresh Request                  | 9 (highest)    |
| Port 0 Read Channel Urgent Request  | 8              |
| Port 0 Write Channel Urgent Request | 7              |
| Port 1 Read Channel Urgent Request  | 6              |
| Port 1 Write Channel Urgent Request | 5              |
| Port 0 Read Channel Normal Request  | 4              |
| Port 0 Write Channel Normal Request | 3              |
| Port 1 Read Channel Normal Request  | 2              |
| Port 1 Write Channel Normal Request | 1 (lowest)     |

## L2 Memory Controller Event Control

The following sections describe L2 memory controller event control features, such as error response.

A bus error is signaled under any of the following conditions.

- A write access to ROM address space
- A read/write access to reserved address space
- A write access (including a TESTSET) to a restricted memory bank
- An ECC multi-bit error in an ECC enabled bank. A non-modulo32-bit write to an ECC enabled bank can also potentially create a bus error response due to an ECC multi-bit error. This is because the L2 controller implements a 32-bit ECC, and therefore a non-modulo32-bit write results in a read. This read may create multi-bit errors even if the memory was initialized.

Bus error notifications are stored in the L2CTL\_STAT register, and addresses that generated the error on a given port are stored in that port's L2CTL\_EADDRx register. The details of the error are stored in a port's L2CTL\_ETx register.

## L2CTL Register Descriptions

L2 Memory Controller (L2CTL) contains the following registers.

**Table 10-5:** L2CTL Module Registers

| Name           | Description                    |
|----------------|--------------------------------|
| L2CTL_CTL      | Control Register               |
| L2CTL_ACTL_C0  | Access Control Core 0 Register |
| L2CTL_ACTL_C1  | Access Control Core 1 Register |
| L2CTL_ACTL_SYS | Access Control System Register |
| L2CTL_STAT     | Status Register                |
| L2CTL_RPCR     | Read Priority Count Register   |
| L2CTL_WPCR     | Write Priority Count Register  |
| L2CTL_RFA      | Refresh Address Register       |
| L2CTL_ERRADDR0 | ECC Error Address 0 Register   |
| L2CTL_ERRADDR1 | ECC Error Address 1 Register   |
| L2CTL_ERRADDR2 | ECC Error Address 2 Register   |
| L2CTL_ERRADDR3 | ECC Error Address 3 Register   |
| L2CTL_ERRADDR4 | ECC Error Address 4 Register   |
| L2CTL_ERRADDR5 | ECC Error Address 5 Register   |
| L2CTL_ERRADDR6 | ECC Error Address 6 Register   |
| L2CTL_ERRADDR7 | ECC Error Address 7 Register   |
| L2CTL_ET0      | Error Type 0 Register          |

Table 10-5: L2CTL Module Registers (Continued)

| Name         | Description                   |
|--------------|-------------------------------|
| L2CTL_EADDR0 | Error Type 0 Address Register |
| L2CTL_ET1    | Error Type 1 Register         |
| L2CTL_EADDR1 | Error Type 1 Address Register |

Control Register

The L2CTL\_CTL register includes a write protection bit, enables L2 banks, and selects mapping of banks (as ECC RAM or data RAM).

L2CTL\_CTL: Control Register - R/W

Reset = 0x0000 0000

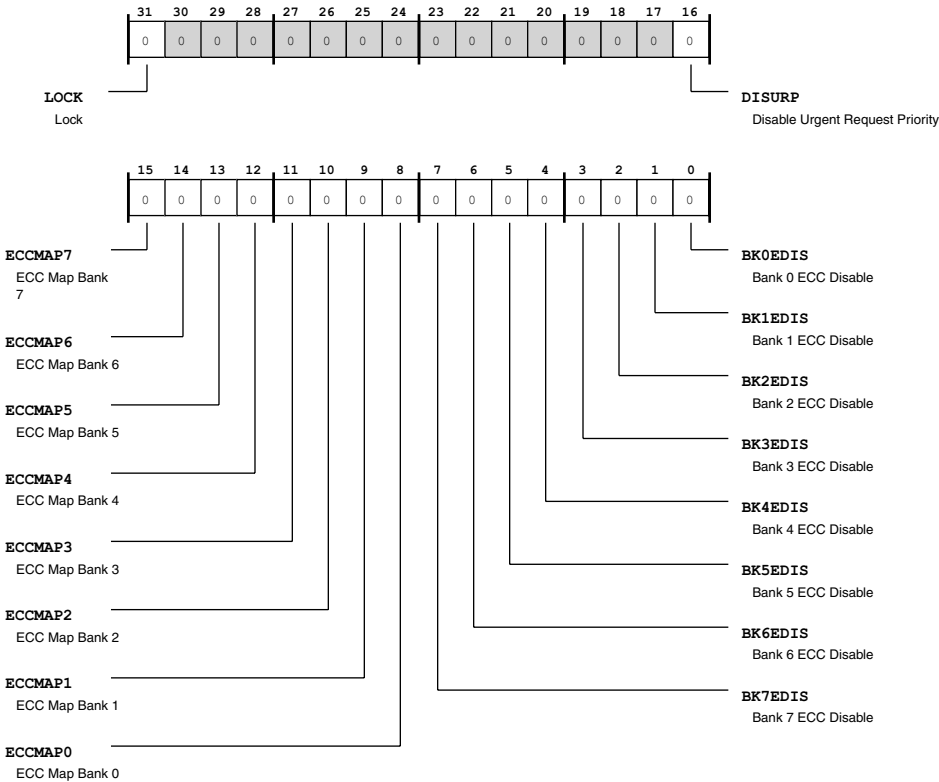


Figure 10-3: L2CTL\_CTL Register Diagram



**Table 10-6:** L2CTL\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the L2CTL_CTL.LOCK bit is set, the L2CTL_CTL register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                 |
|         |          | 1                      Lock                                                                                                                   |
| 16      | DISURP   | Disable Urgent Request Priority.<br>The L2CTL_CTL.DISURP disables urgent request priority mode for all L2 banks.                              |
|         |          | 0                      Enable URP                                                                                                             |
|         |          | 1                      Disable URP                                                                                                            |
| 15      | ECCMAP7  | ECC Map Bank 7.<br>The L2CTL_CTL.ECCMAP7 bit selects whether L2 bank 7 addresses ECC RAM or data RAM.                                         |
|         |          | 0                      Data RAM                                                                                                               |
|         |          | 1                      ECC RAM                                                                                                                |
| 14      | ECCMAP6  | ECC Map Bank 6.<br>The L2CTL_CTL.ECCMAP6 bit selects whether L2 bank 6 addresses ECC RAM or data RAM.                                         |
|         |          | 0                      Data RAM                                                                                                               |
|         |          | 1                      ECC RAM                                                                                                                |
| 13      | ECCMAP5  | ECC Map Bank 5.<br>The L2CTL_CTL.ECCMAP5 bit selects whether L2 bank 5 addresses ECC RAM or data RAM.                                         |
|         |          | 0                      Data RAM                                                                                                               |
|         |          | 1                      ECC RAM                                                                                                                |

**Table 10-6:** L2CTL\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 12      | ECCMAP4  | ECC Map Bank 4.<br>The L2CTL_CTL.ECCMAP4 bit selects whether L2 bank 4 addresses ECC RAM or data RAM. |
|         |          | 0 Data RAM                                                                                            |
|         |          | 1 ECC RAM                                                                                             |
| 11      | ECCMAP3  | ECC Map Bank 3.<br>The L2CTL_CTL.ECCMAP3 bit selects whether L2 bank 3 addresses ECC RAM or data RAM. |
|         |          | 0 Data RAM                                                                                            |
|         |          | 1 ECC RAM                                                                                             |
| 10      | ECCMAP2  | ECC Map Bank 2.<br>The L2CTL_CTL.ECCMAP2 bit selects whether L2 bank 2 addresses ECC RAM or data RAM. |
|         |          | 0 Data RAM                                                                                            |
|         |          | 1 ECC RAM                                                                                             |
| 9       | ECCMAP1  | ECC Map Bank 1.<br>The L2CTL_CTL.ECCMAP1 bit selects whether L2 bank 1 addresses ECC RAM or data RAM. |
|         |          | 0 Data RAM                                                                                            |
|         |          | 1 ECC RAM                                                                                             |
| 8       | ECCMAP0  | ECC Map Bank 0.<br>The L2CTL_CTL.ECCMAP0 bit selects whether L2 bank 0 addresses ECC RAM or data RAM. |
|         |          | 0 Data RAM                                                                                            |
|         |          | 1 ECC RAM                                                                                             |

**Table 10-6:** L2CTL\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                            |
|---------|----------|------------------------------------------------------------------------------------|
| 7       | BK7EDIS  | Bank 7 ECC Disable.<br>The L2CTL_CTL.BK7EDIS bit disables L2 bank 7 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |
| 6       | BK6EDIS  | Bank 6 ECC Disable.<br>The L2CTL_CTL.BK6EDIS bit disables L2 bank 6 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |
| 5       | BK5EDIS  | Bank 5 ECC Disable.<br>The L2CTL_CTL.BK5EDIS bit disables L2 bank 5 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |
| 4       | BK4EDIS  | Bank 4 ECC Disable.<br>The L2CTL_CTL.BK4EDIS bit disables L2 bank 4 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |
| 3       | BK3EDIS  | Bank 3 ECC Disable.<br>The L2CTL_CTL.BK3EDIS bit disables L2 bank 3 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |
| 2       | BK2EDIS  | Bank 2 ECC Disable.<br>The L2CTL_CTL.BK2EDIS bit disables L2 bank 2 ECC operation. |
|         |          | 0      Enable ECC                                                                  |
|         |          | 1      Disable ECC                                                                 |

Table 10-6: L2CTL\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                            |
|---------|----------|------------------------------------------------------------------------------------|
| 1       | BK1EDIS  | Bank 1 ECC Disable.<br>The L2CTL_CTL.BK1EDIS bit disables L2 bank 1 ECC operation. |
|         |          | 0                      Enable ECC                                                  |
|         |          | 1                      Disable ECC                                                 |
| 0       | BK0EDIS  | Bank 0 ECC Disable.<br>The L2CTL_CTL.BK0EDIS bit disables L2 bank 0 ECC operation. |
|         |          | 0                      Enable ECC                                                  |
|         |          | 1                      Disable ECC                                                 |

Access Control Core 0 Register

The L2CTL\_ACTL\_C0 register includes a write protection bit and enables core 0 write access for L2 banks.

L2CTL\_ACTL\_C0: Access Control Core 0 Register - R/W

Reset = 0x0000 0000

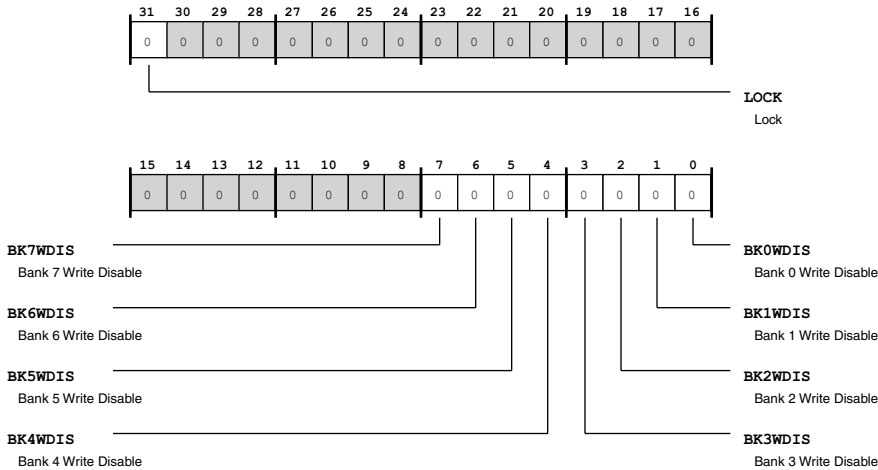


Figure 10-4: L2CTL\_ACTL\_C0 Register Diagram

**Table 10-7:** L2CTL\_ACTL\_C0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the L2CTL_ACTL_C0.LOCK bit is set, the L2CTL_ACTL_C0 register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                         |
|         |          | 1                      Lock                                                                                                                           |
| 7       | BK7WDIS  | Bank 7 Write Disable.<br>The L2CTL_ACTL_C0.BK7WDIS bit disables core 0 writes to L2 bank 7 RAM.                                                       |
|         |          | 0                      Enable Write                                                                                                                   |
|         |          | 1                      Disable Write                                                                                                                  |
| 6       | BK6WDIS  | Bank 6 Write Disable.<br>The L2CTL_ACTL_C0.BK6WDIS bit disables core 0 writes to L2 bank 6 RAM.                                                       |
|         |          | 0                      Enable Write                                                                                                                   |
|         |          | 1                      Disable Write                                                                                                                  |
| 5       | BK5WDIS  | Bank 5 Write Disable.<br>The L2CTL_ACTL_C0.BK5WDIS bit disables core 0 writes to L2 bank 5 RAM.                                                       |
|         |          | 0                      Enable Write                                                                                                                   |
|         |          | 1                      Disable Write                                                                                                                  |
| 4       | BK4WDIS  | Bank 4 Write Disable.<br>The L2CTL_ACTL_C0.BK4WDIS bit disables core 0 writes to L2 bank 4 RAM.                                                       |
|         |          | 0                      Enable Write                                                                                                                   |
|         |          | 1                      Disable Write                                                                                                                  |

**Table 10-7:** L2CTL\_ACTL\_C0 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 3       | BK3WDIS  | Bank 3 Write Disable.<br>The L2CTL_ACTL_C0.BK3WDIS bit disables core 0 writes to L2 bank 3 RAM. |
|         |          | 0      Enable Write                                                                             |
|         |          | 1      Disable Write                                                                            |
| 2       | BK2WDIS  | Bank 2 Write Disable.<br>The L2CTL_ACTL_C0.BK2WDIS bit disables core 0 writes to L2 bank 2 RAM. |
|         |          | 0      Enable Write                                                                             |
|         |          | 1      Disable Write                                                                            |
| 1       | BK1WDIS  | Bank 1 Write Disable.<br>The L2CTL_ACTL_C0.BK1WDIS bit disables core 0 writes to L2 bank 1 RAM. |
|         |          | 0      Enable Write                                                                             |
|         |          | 1      Disable Write                                                                            |
| 0       | BK0WDIS  | Bank 0 Write Disable.<br>The L2CTL_ACTL_C0.BK0WDIS bit disables core 0 writes to L2 bank 0 RAM. |
|         |          | 0      Enable Write                                                                             |
|         |          | 1      Disable Write                                                                            |

## Access Control Core 1 Register

The L2CTL\_ACTL\_C1 register includes a write protection bit and enables core 1 write access for L2 banks.

**L2CTL\_ACTL\_C1: Access Control Core 1 Register - R/W**

Reset = 0x0000 0000

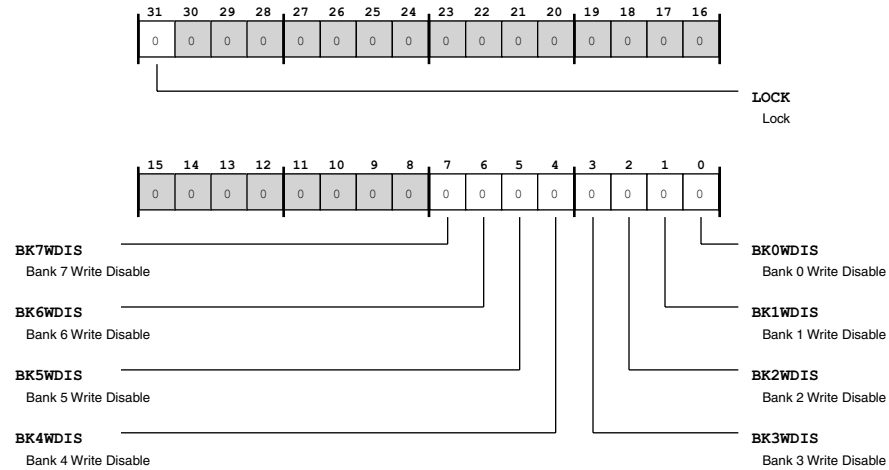


**Figure 10-5:** L2CTL\_ACTL\_C1 Register Diagram

**Table 10-8:** L2CTL\_ACTL\_C1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the L2CTL_ACTL_C1 . LOCK bit is set, the L2CTL_ACTL_C1 register is read only (locked). |
|         |          | 0      Unlock                                                                                                                                           |
|         |          | 1      Lock                                                                                                                                             |
| 7       | BK7WDIS  | Bank 7 Write Disable.<br>The L2CTL_ACTL_C1 . BK7WDIS bit disables core 1 writes to L2 bank 7 RAM.                                                       |
|         |          | 0      Enable Write                                                                                                                                     |
|         |          | 1      Disable Write                                                                                                                                    |

**Table 10-8:** L2CTL\_ACTL\_C1 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 6       | BK6WDIS  | Bank 6 Write Disable.<br>The L2CTL_ACTL_C1.BK6WDIS bit disables core 1 writes to L2 bank 6 RAM. |
|         |          | 0 Enable Write                                                                                  |
|         |          | 1 Disable Write                                                                                 |
| 5       | BK5WDIS  | Bank 5 Write Disable.<br>The L2CTL_ACTL_C1.BK5WDIS bit disables core 1 writes to L2 bank 5 RAM. |
|         |          | 0 Enable Write                                                                                  |
|         |          | 1 Disable Write                                                                                 |
| 4       | BK4WDIS  | Bank 4 Write Disable.<br>The L2CTL_ACTL_C1.BK4WDIS bit disables core 1 writes to L2 bank 4 RAM. |
|         |          | 0 Enable Write                                                                                  |
|         |          | 1 Disable Write                                                                                 |
| 3       | BK3WDIS  | Bank 3 Write Disable.<br>The L2CTL_ACTL_C1.BK3WDIS bit disables core 1 writes to L2 bank 3 RAM. |
|         |          | 0 Enable Write                                                                                  |
|         |          | 1 Disable Write                                                                                 |
| 2       | BK2WDIS  | Bank 2 Write Disable.<br>The L2CTL_ACTL_C1.BK2WDIS bit disables core 1 writes to L2 bank 2 RAM. |
|         |          | 0 Enable Write                                                                                  |
|         |          | 1 Disable Write                                                                                 |



**Table 10-8:** L2CTL\_ACTL\_C1 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 1       | BK1WDIS  | Bank 1 Write Disable.<br>The L2CTL_ACTL_C1.BK1WDIS bit disables core 1 writes to L2 bank 1 RAM. |
|         |          | 0                      Enable Write                                                             |
|         |          | 1                      Disable Write                                                            |
| 0       | BK0WDIS  | Bank 0 Write Disable.<br>The L2CTL_ACTL_C1.BK0WDIS bit disables core 1 writes to L2 bank 0 RAM. |
|         |          | 0                      Enable Write                                                             |
|         |          | 1                      Disable Write                                                            |

## Access Control System Register

The L2CTL\_ACTL\_SYS register includes a write protection bit and enables system and DMA write access for L2 banks.

### L2CTL\_ACTL\_SYS: Access Control System Register - R/W

Reset = 0x0000 0000

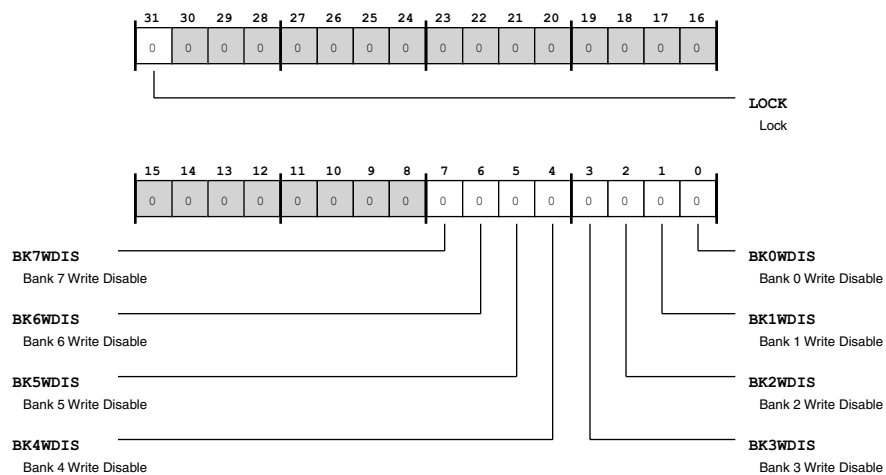


**Figure 10-6:** L2CTL\_ACTL\_SYS Register Diagram

**Table 10-9:** L2CTL\_ACTL\_SYS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the L2CTL_ACTL_SYS.LOCK bit is set, the L2CTL_ACTL_SYS register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                           |
|         |          | 1                      Lock                                                                                                                             |
| 7       | BK7WDIS  | Bank 7 Write Disable.<br>The L2CTL_ACTL_SYS.BK7WDIS bit disables system or DMA writes to L2 bank 7 RAM.                                                 |
|         |          | 0                      Enable Write                                                                                                                     |
|         |          | 1                      Disable Write                                                                                                                    |
| 6       | BK6WDIS  | Bank 6 Write Disable.<br>The L2CTL_ACTL_SYS.BK6WDIS bit disables system or DMA writes to L2 bank 6 RAM.                                                 |
|         |          | 0                      Enable Write                                                                                                                     |
|         |          | 1                      Disable Write                                                                                                                    |
| 5       | BK5WDIS  | Bank 5 Write Disable.<br>The L2CTL_ACTL_SYS.BK5WDIS bit disables system or DMA writes to L2 bank 5 RAM.                                                 |
|         |          | 0                      Enable Write                                                                                                                     |
|         |          | 1                      Disable Write                                                                                                                    |
| 4       | BK4WDIS  | Bank 4 Write Disable.<br>The L2CTL_ACTL_SYS.BK4WDIS bit disables system or DMA writes to L2 bank 4 RAM.                                                 |
|         |          | 0                      Enable Write                                                                                                                     |
|         |          | 1                      Disable Write                                                                                                                    |

**Table 10-9:** L2CTL\_ACTL\_SYS Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------|
| 3       | BK3WDIS  | Bank 3 Write Disable.<br>The L2CTL_ACTL_SYS.BK3WDIS bit disables system or DMA writes to L2 bank 3 RAM. |
|         |          | 0      Enable Write                                                                                     |
|         |          | 1      Disable Write                                                                                    |
| 2       | BK2WDIS  | Bank 2 Write Disable.<br>The L2CTL_ACTL_SYS.BK2WDIS bit disables system or DMA writes to L2 bank 2 RAM. |
|         |          | 0      Enable Write                                                                                     |
|         |          | 1      Disable Write                                                                                    |
| 1       | BK1WDIS  | Bank 1 Write Disable.<br>The L2CTL_ACTL_SYS.BK1WDIS bit disables system or DMA writes to L2 bank 1 RAM. |
|         |          | 0      Enable Write                                                                                     |
|         |          | 1      Disable Write                                                                                    |
| 0       | BK0WDIS  | Bank 0 Write Disable.<br>The L2CTL_ACTL_SYS.BK0WDIS bit disables system or DMA writes to L2 bank 0 RAM. |
|         |          | 0      Enable Write                                                                                     |
|         |          | 1      Disable Write                                                                                    |

## Status Register

The L2CTL\_STAT register indicates ECC error status, refresh register status, and bus error status.

L2CTL\_STAT: Status Register - R/W

Figure 10-7: L2CTL\_STAT Register Diagram

Table 10-10: L2CTL\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                           |                      |
|---------|----------|-------------------------------------------------------------------------------------------------------------------|----------------------|
| 15      | ECCERR7  | ECC Error Bank 7.<br>The L2CTL_STAT.ECCERR7 bit indicates that an ECC double bit error occurred inside L2 bank 7. |                      |
|         |          | 0                                                                                                                 | No Status            |
|         |          | 1                                                                                                                 | ECC Double Bit Error |
| 14      | ECCERR6  | ECC Error Bank 6.<br>The L2CTL_STAT.ECCERR6 bit indicates that an ECC double bit error occurred inside L2 bank 6. |                      |
|         |          | 0                                                                                                                 | No Status            |
|         |          | 1                                                                                                                 | ECC Double Bit Error |

**Table 10-10:** L2CTL\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------|
| 13      | ECCERR5  | ECC Error Bank 5.<br>The L2CTL_STAT.ECCERR5 bit indicates that an ECC double bit error occurred inside L2 bank 5. |
|         |          | 0 No Status                                                                                                       |
|         |          | 1 ECC Double Bit Error                                                                                            |
| 12      | ECCERR4  | ECC Error Bank 4.<br>The L2CTL_STAT.ECCERR4 bit indicates that an ECC double bit error occurred inside L2 bank 4. |
|         |          | 0 No Status                                                                                                       |
|         |          | 1 ECC Double Bit Error                                                                                            |
| 11      | ECCERR3  | ECC Error Bank 3.<br>The L2CTL_STAT.ECCERR3 bit indicates that an ECC double bit error occurred inside L2 bank 3. |
|         |          | 0 No Status                                                                                                       |
|         |          | 1 ECC Double Bit Error                                                                                            |
| 10      | ECCERR2  | ECC Error Bank 2.<br>The L2CTL_STAT.ECCERR2 bit indicates that an ECC double bit error occurred inside L2 bank 2. |
|         |          | 0 No Status                                                                                                       |
|         |          | 1 ECC Double Bit Error                                                                                            |
| 9       | ECCERR1  | ECC Error Bank 1.<br>The L2CTL_STAT.ECCERR1 bit indicates that an ECC double bit error occurred inside L2 bank 1. |
|         |          | 0 No Status                                                                                                       |
|         |          | 1 ECC Double Bit Error                                                                                            |

**Table 10-10:** L2CTL\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | ECCERR0  | ECC Error Bank 0.<br>The L2CTL_STAT.ECCERR0 bit indicates that an ECC double bit error occurred inside L2 bank 0.                                       |
|         |          | 0                      No Status                                                                                                                        |
|         |          | 1                      ECC Double Bit Error                                                                                                             |
| 4       | RFRS     | Refresh Register Status.<br>The L2CTL_STAT.RFRS bit indicates whether a refresh request is pending (in progress) or that there are no pending requests. |
|         |          | 0                      No Pending Requests                                                                                                              |
|         |          | 1                      Request Pending (Refresh in Progress)                                                                                            |
| 1       | ERR1     | Error Port 1.<br>The L2CTL_STAT.ERR1 indicates whether the L2CTL has detected a bus access error on L2's bus port 1.                                    |
|         |          | 0                      No Error                                                                                                                         |
|         |          | 1                      Bus Access Error                                                                                                                 |
| 0       | ERR0     | Error Port 0.<br>The L2CTL_STAT.ERR0 indicates whether the L2CTL has detected a bus access error on L2's bus port 0.                                    |
|         |          | 0                      No Error                                                                                                                         |
|         |          | 1                      Bus Access Error                                                                                                                 |

## Read Priority Count Register

The L2CTL\_RPCR register stores the count value to be used for priority elevation for bus read channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of L2CLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus read channel. A best practice is to program this register before initiating an L2 access.

**L2CTL\_RPCR: Read Priority Count Register - R/W**

Reset = 0x0000 0f0f

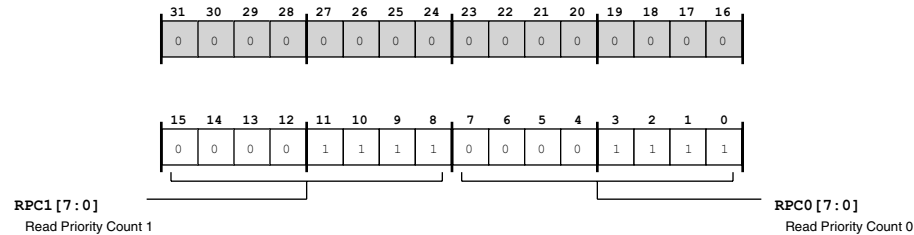


**Figure 10-8:** L2CTL\_RPCR Register Diagram

**Table 10-11:** L2CTL\_RPCR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 15:8    | RPC1     | Read Priority Count 1.<br>The L2CTL_RPCR.RPC1 bits hold the priority count for L2 bus read channel 1. |
| 7:0     | RPC0     | Read Priority Count 0.<br>The L2CTL_RPCR.RPC0 bits hold the priority count for L2 bus read channel 0. |

## Write Priority Count Register

The L2CTL\_WPCR register stores the count value to be used for priority elevation for bus write channels. If a bus channel is not granted access from the bank arbiter, the channel waits for the programmed number of L2CLK cycles, before the request is elevated to a high priority request. If a priority count value is programmed as zero for a channel, that channel does not raise the urgent priority request.

This is a read/write register, but a new value in the corresponding field must be written only when there are no outstanding transactions on the corresponding bus write channel. A best practice is to program this register before initiating an L2 access.

L2CTL\_WPCR: Write Priority Count Register - R/W

Reset = 0x0000 0f0f

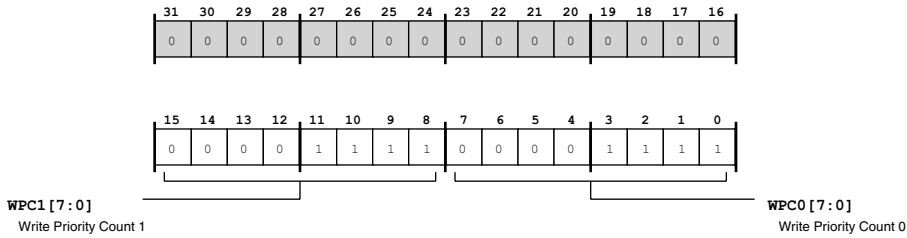


Figure 10-9: L2CTL\_WPCR Register Diagram

Table 10-12: L2CTL\_WPCR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------|
| 15:8    | WPC1     | Write Priority Count 1.<br>The L2CTL_WPCR.WPC1 bits hold the priority count for L2 bus write channel 0. |
| 7:0     | WPC0     | Write Priority Count 0.<br>The L2CTL_WPCR.WPC0 bits hold the priority count for L2 bus write channel 1. |

Refresh Address Register

The L2CTL\_RFA register stores the refresh address value. When this register is written, L2 initiates an atomic read-write operation to the address value written into the register. This is a read/write register, but a new value in the corresponding field has to be written only when there are no outstanding refresh request pending (L2CTL\_STAT.RFRS =0). If a write occurs while a request is pending, the L2CTL generates a bus error, and the write does not take effect.



**L2CTL\_RFA: Refresh Address Register - R/W**

Reset = 0xc808 0000

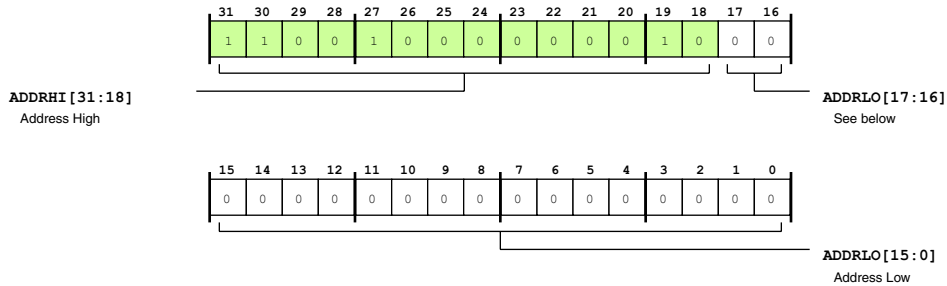


**Figure 10-10:** L2CTL\_RFA Register Diagram

**Table 10-13:** L2CTL\_RFA Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:18   | ADDRHI   | Address High.<br>The L2CTL_RFA.ADDRHI bits hold the high 16-bits of the L2 refresh address. Note that the upper 14 bits are hard-coded to the upper bits of the L2 address map. |
| 17:0    | ADDRLO   | Address Low.<br>The L2CTL_RFA.ADDRLO bits hold the low 16-bits of the L2 refresh address. Note that the lowest three bits are do not care.                                      |

## ECC Error Address 0 Register

The L2CTL\_ERRADDR0 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR0) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR0: ECC Error Address 0 Register - R/NW

Reset = 0xc808 0000

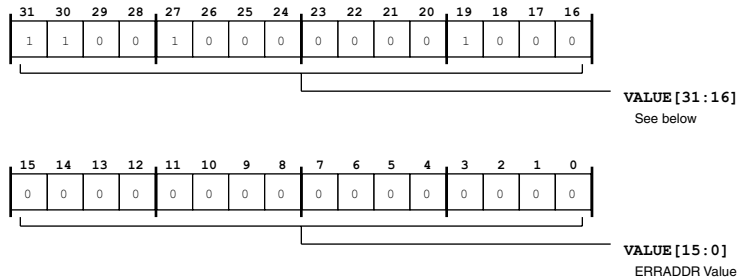


Figure 10-11: L2CTL\_ERRADDR0 Register Diagram

Table 10-14: L2CTL\_ERRADDR0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR0.VALUE bits hold the address containing the ECC double bit error. |

ECC Error Address 1 Register

The L2CTL\_ERRADDR1 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR1) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR1: ECC Error Address 1 Register - R/NW

Reset = 0xc808 0000

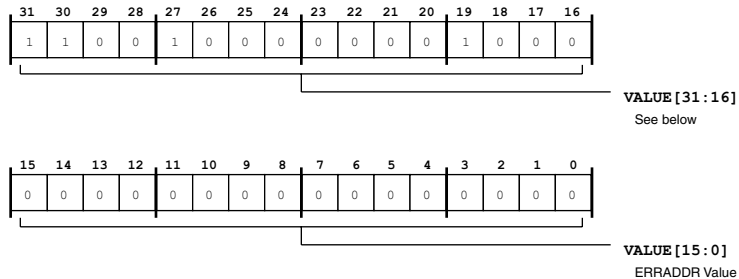


Figure 10-12: L2CTL\_ERRADDR1 Register Diagram

**Table 10-15:** L2CTL\_ERRADDR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR1.VALUE bits hold the address containing the ECC double bit error. |

## ECC Error Address 2 Register

The L2CTL\_ERRADDR2 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR2) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

**L2CTL\_ERRADDR2: ECC Error Address 2 Register - R/NW**

Reset = 0xc808 0000

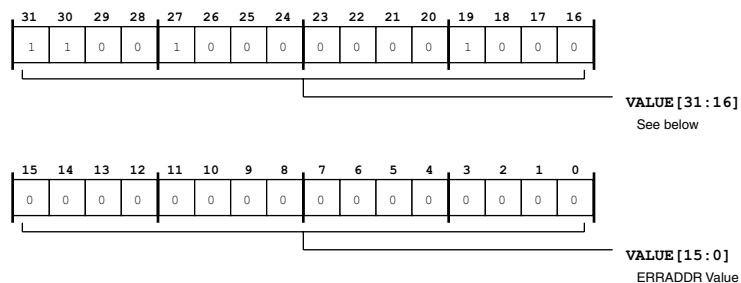


**Figure 10-13:** L2CTL\_ERRADDR2 Register Diagram

**Table 10-16:** L2CTL\_ERRADDR2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR2.VALUE bits hold the address containing the ECC double bit error. |

## ECC Error Address 3 Register

The L2CTL\_ERRADDR3 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR3) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR3: ECC Error Address 3 Register - R/NW

Reset = 0xc808 0000

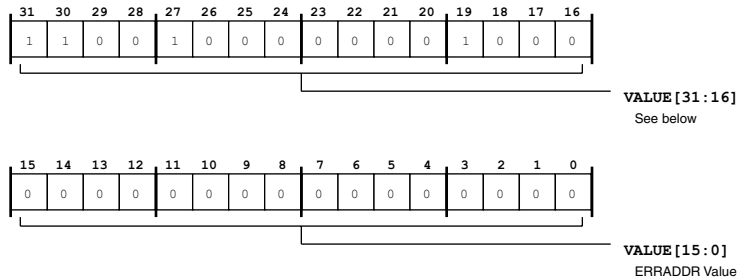


Figure 10-14: L2CTL\_ERRADDR3 Register Diagram

Table 10-17: L2CTL\_ERRADDR3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR3.VALUE bits hold the address containing the ECC double bit error. |

ECC Error Address 4 Register

The L2CTL\_ERRADDR4 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR4) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR4: ECC Error Address 4 Register - R/NW

Reset = 0xc808 0000

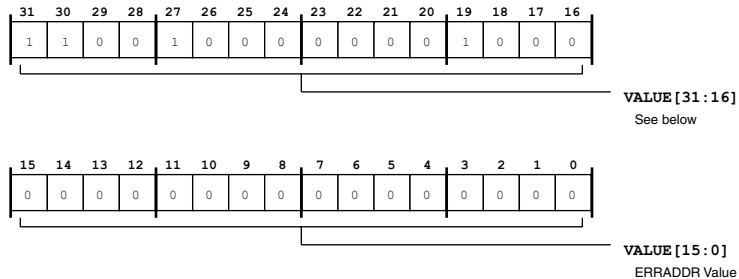


Figure 10-15: L2CTL\_ERRADDR4 Register Diagram

**Table 10-18:** L2CTL\_ERRADDR4 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR4.VALUE bits hold the address containing the ECC double bit error. |

## ECC Error Address 5 Register

The L2CTL\_ERRADDR5 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR5) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

**L2CTL\_ERRADDR5: ECC Error Address 5 Register - R/NW**

Reset = 0xc808 0000

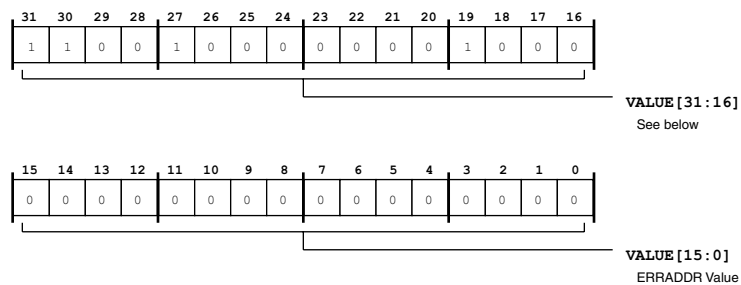


**Figure 10-16:** L2CTL\_ERRADDR5 Register Diagram

**Table 10-19:** L2CTL\_ERRADDR5 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR5.VALUE bits hold the address containing the ECC double bit error. |

## ECC Error Address 6 Register

The L2CTL\_ERRADDR6 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR6) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR6: ECC Error Address 6 Register - R/NW

Reset = 0xc808 0000

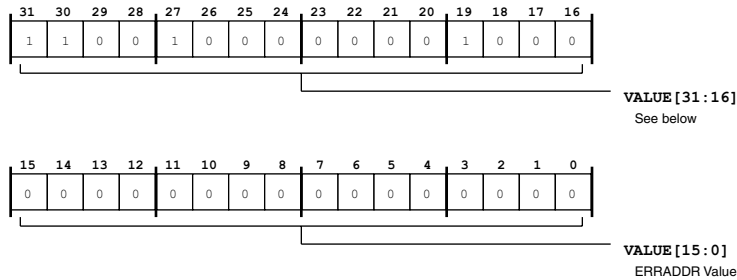


Figure 10-17: L2CTL\_ERRADDR6 Register Diagram

Table 10-20: L2CTL\_ERRADDR6 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR6.VALUE bits hold the address containing the ECC double bit error. |

ECC Error Address 7 Register

The L2CTL\_ERRADDR7 register holds the address containing an ECC multi-bit error for the corresponding bank. The L2CTL updates this register only if the bank's status bit (L2CTL\_STAT.ECCERR7) is cleared. After the bank's status bit is set for an error, further errors in the same bank are not detected until a W1C clears the status bit.

L2CTL\_ERRADDR7: ECC Error Address 7 Register - R/NW

Reset = 0xc808 0000

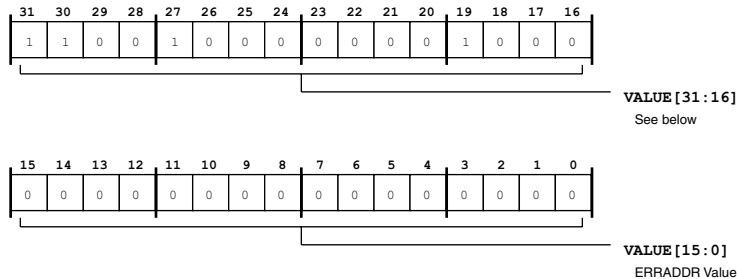


Figure 10-18: L2CTL\_ERRADDR7 Register Diagram

**Table 10-21:** L2CTL\_ERRADDR7 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_ERRADDR7.VALUE bits hold the address containing the ECC double bit error. |

## Error Type 0 Register

The L2CTL\_ET0 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port. This register is updated only if the corresponding error status bit L2CTL\_STAT.ERR0 is cleared. After the status bit is set for an error, further errors do not update the L2CTL\_ET0 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the L2CTL\_ET0 captures the write access error, keeping in sync with the error address register (L2CTL\_EADDR0).

L2CTL\_ET0: Error Type 0 Register - R/NW

Reset = 0x0000 0000

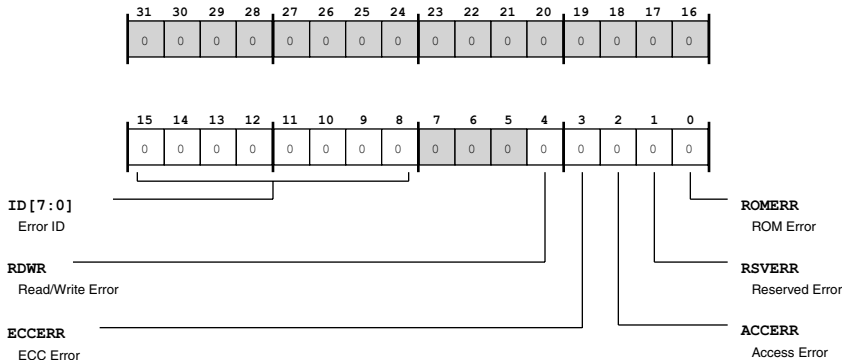


Figure 10-19: L2CTL\_ET0 Register Diagram

Table 10-22: L2CTL\_ET0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------|
| 15:8    | ID       | Error ID.<br>The L2CTL_ET0.ID bits hold the bus master ID of the access that caused error.            |
| 4       | RDWR     | Read/Write Error.<br>The L2CTL_ET0.RDWR bit indicates whether a read or write access caused error.    |
|         | 0        | Read Access created Error                                                                             |
|         | 1        | Write Access created Error                                                                            |
| 3       | ECCERR   | ECC Error.<br>The L2CTL_ET0.ECCERR bit indicates whether the access had an ECC double bit error.      |
| 2       | ACCERR   | Access Error.<br>The L2CTL_ET0.ACCERR bit indicates whether the access went to a restricted bank.     |
| 1       | RSVERR   | Reserved Error.<br>The L2CTL_ET0.RSVERR bit indicates whether the access went to a reserved location. |



**Table 10-22:** L2CTL\_ET0 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------|
| 0       | ROMERR   | ROM Error.<br>The L2CTL_ET0.ROMERR bit indicates whether a write access went to a ROM area. |

## Error Type 0 Address Register

The L2CTL\_EADDR0 register holds the address that created an access error on the L2 port 0 bus interface. This register is be updated only if the corresponding error status bit (L2CTL\_STAT.ERR0) is cleared. After the status bit is set for an error, further errors do not update the L2CTL\_EADDR0 register until a WIC clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

**L2CTL\_EADDR0: Error Type 0 Address Register - R/NW**

Reset = 0xc800 0000

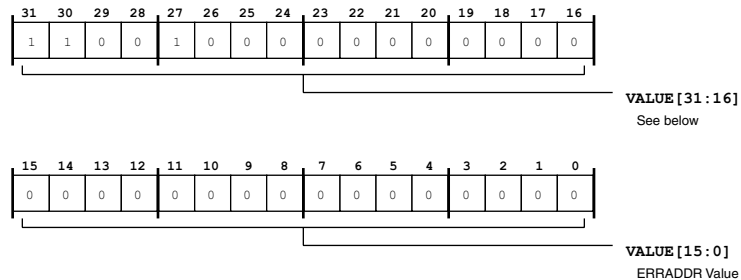


**Figure 10-20:** L2CTL\_EADDR0 Register Diagram

**Table 10-23:** L2CTL\_EADDR0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                               |
|---------|----------|---------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_EADDR0.VALUE bits hold the address causing the bus error. |

## Error Type 1 Register

The L2CTL\_ET1 register holds information about the error transaction that has occurred on the bus for the corresponding L2 bus port. This register is updated only if the corresponding error status bit L2CTL\_STAT.ERR1 is cleared. After the status bit is set for an error, further errors do not update the L2CTL\_ET1 register until a WIC clears the corresponding status bit. If read and write access errors occur

simultaneously, the L2CTL\_ET1 captures the write access error, keeping in sync with the error address register (L2CTL\_EADDR1).

L2CTL\_ET1: Error Type 1 Register - R/NW

Reset = 0x0000 0000

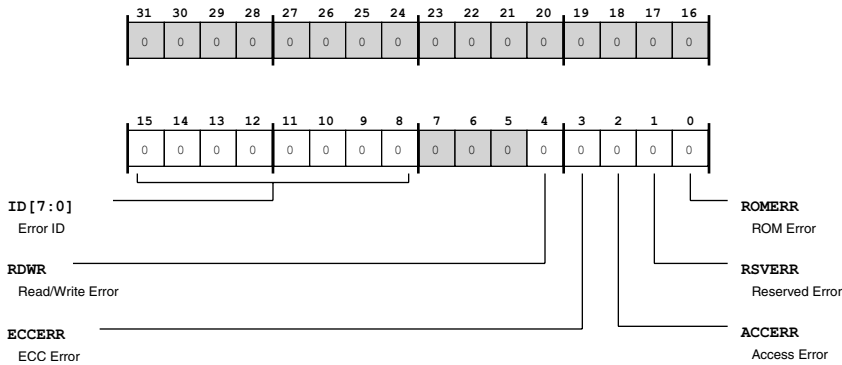


Figure 10-21: L2CTL\_ET1 Register Diagram

Table 10-24: L2CTL\_ET1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------|
| 15:8    | ID       | Error ID.<br>The L2CTL_ET1 . ID bits hold the bus master ID of the access that caused error.            |
| 4       | RDWR     | Read/Write Error.<br>The L2CTL_ET1 . RDWR bit indicates whether a read or write access caused error.    |
|         |          | 0      Read Access created Error                                                                        |
|         |          | 1      Write Access created Error                                                                       |
| 3       | ECCERR   | ECC Error.<br>The L2CTL_ET1 . ECCERR bit indicates whether the access had an ECC double bit error.      |
| 2       | ACCERR   | Access Error.<br>The L2CTL_ET1 . ACCERR bit indicates whether the access went to a restricted bank.     |
| 1       | RSVERR   | Reserved Error.<br>The L2CTL_ET1 . RSVERR bit indicates whether the access went to a reserved location. |

**Table 10-24:** L2CTL\_ET1 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------|
| 0       | ROMERR   | ROM Error.<br>The L2CTL_ET1.ROMERR bit indicates whether a write access went to a ROM area. |

## Error Type 1 Address Register

The L2CTL\_EADDR1 register holds the address that created an access error on the L2 port 0 bus interface. This register is be updated only if the corresponding error status bit (L2CTL\_STAT.ERR1) is cleared. After the status bit is set for an error, further errors do not update the L2CTL\_EADDR1 register until a W1C clears the corresponding status bit. If read and write access errors occur simultaneously, the register captures the write access error address.

**L2CTL\_EADDR1: Error Type 1 Address Register - R/NW**

Reset = 0xc800 0000

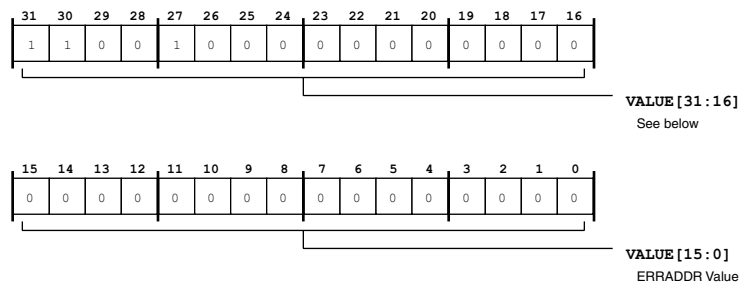


**Figure 10-22:** L2CTL\_EADDR1 Register Diagram

**Table 10-25:** L2CTL\_EADDR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                               |
|---------|----------|---------------------------------------------------------------------------------------|
| 31:0    | VALUE    | ERRADDR Value.<br>The L2CTL_EADDR1.VALUE bits hold the address causing the bus error. |

## BF60x-Specific Information

The L2CLK runs at the same rate as SYSCLK.

The L2 memory subsystem has 2 SCB ports accessing 256 KB of RAM in 8 banks and 32 KB of ROM in a single bank.

## L2 Memory Controller ADSP-BF60x Throughput

**Table 10-26:** ADSP-BF60x Throughput

| Master     | Condition                                                                                                                                                                                                                                                                                                                                 | Core Throughput (MBPS) | DMA Throughput (MBPS) |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|-----------------------|
| Core Read  | Data Cache – burst read (cache fill) single core, data cache enabled. Instruction execution is from L1. Cache fill results in L2 burst access.                                                                                                                                                                                            | 540                    | N/A                   |
| Core Read  | Data Cache – dual burst read single core (with port preference) (cache fill).                                                                                                                                                                                                                                                             | 940                    | N/A                   |
| Core Read  | Instruction Cache – burst read, single core. Execution from L2 with instruction caching enabled.                                                                                                                                                                                                                                          | 550                    | N/A                   |
| Core Read  | Instruction Cache and data cache enabled – burst read, single core. Execution from L2 with data access also from L2. Here the instruction and data use must be aligned in such a way that an instruction miss is immediately followed by a data cache miss in the same instruction. This results in back-to-back burst read access to L2. | 666                    | N/A                   |
| Core Write | Data Cache – burst write (write back), single core, cache enabled                                                                                                                                                                                                                                                                         | 1028                   | N/A                   |
| Core Write | Non data cache – single write, cache disabled, single writes to L2, single core.<br>(1000 MBPS achieved by running in higher interrupt level)                                                                                                                                                                                             | 800, 1000              | N/A                   |
| DMA Read   | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 8                                                                                                                                                                                                                                        | N/A                    | 880                   |
| DMA Read   | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 4                                                                                                                                                                                                                                        | N/A                    | 800                   |
| DMA Read   | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 2                                                                                                                                                                                                                                        | N/A                    | 666                   |
| DMA Write  | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 8                                                                                                                                                                                                                                        | N/A                    | 880                   |

**Table 10-26:** ADSP-BF60x Throughput (Continued)

| Master    | Condition                                                                                                 | Core Throughput (MBPS) | DMA Throughput (MBPS) |
|-----------|-----------------------------------------------------------------------------------------------------------|------------------------|-----------------------|
| DMA Write | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 4        | N/A                    | 800                   |
| DMA Write | No conflict in the memory bank, multiple DMAs initiated to use the same bank, DDE burst length = 2        | N/A                    | 666                   |
| Core/DMA  | Core and DMA reading from the same memory bank with RPC0/WPC0 = 0 and RPC1/WPC1 = 1, DDE burst length = 8 | 1070                   | 885                   |
| Core/DMA  | Core and DMA writing to the same memory bank with RPC0/WPC0 = 0 and RPC1/WPC1 = 1, DDE burst length = 8   | 1340                   | 670                   |



# 11 Dynamic Memory Controller (DMC)

The dynamic memory controller (DMC) provides a glueless interface between DDR2/LPDDR SDRAMs and the system crossbar interface (SCB). The DMC enables execution of instructions from, as well as transfer of data to and from, DDR2 SDRAM as well as LPDDR SDRAM.

The DMC is partitioned in a manner that allows reconfiguration and maintainability. The memory access protocol state machine along with JEDEC standard specific logic is embedded in the “protocol controller”. An access and operation re-ordering mechanism is incorporated as an “efficiency controller”. An SCB slave interface is provided to interface with the on-chip interconnect. This interface results in an efficient slave implementation owing to its out-of-order transaction capabilities. The control and status registers present in the DMC controller can be accessed using the MMR access bus.

The DMC supports access to the external memory by core and DMA accesses. The external memory address space is divided into four banks.

## DMC Register List

The double data rate-synchronous DRAM (DMC) module provides an interface to external SDRAM. This interface support DDR2 and LPDDR operations. A set of registers govern DMC operations. For more information on DMC functionality, see the DMC register descriptions.

**Table 11-1:** DMC Module Registers

| Name        | Description                 |
|-------------|-----------------------------|
| DMC_CTL     | Control Register            |
| DMC_STAT    | Status Register             |
| DMC_EFFCTL  | Efficiency Control Register |
| DMC_PRI0    | Priority ID Register        |
| DMC_PRIOMSK | Priority ID Mask Register   |
| DMC_CFG     | Configuration Register      |
| DMC_TR0     | Timing 0 Register           |
| DMC_TR1     | Timing 1 Register           |

**Table 11-1:** DMC Module Registers (Continued)

| Name         | Description                          |
|--------------|--------------------------------------|
| DMC_TR2      | Timing 2 Register                    |
| DMC_MSK      | Mask (Mode Register Shadow) Register |
| DMC_MR       | Shadow MR Register                   |
| DMC_EMR1     | Shadow EMR1 Register                 |
| DMC_EMR2     | Shadow EMR2 Register                 |
| DMC_EMR3     | Shadow EMR3 Register                 |
| DMC_DLLCTL   | DLL Control Register                 |
| DMC_PHY_CTL1 | PHY Control 1 Register               |
| DMC_PHY_CTL3 | PHY Control 3 Register               |
| DMC_PADCTL   | PAD Control Register                 |

## ADSP-BF609 specific Register and bit settings

1. Bits 6, 7, 24, and 26 of the DMC\_PHY\_CTL3 register all need to be set to 1.
2. DMC\_DLLCTL register needs to be set to 0x54B. This keeps the default reset values and programs DATACYC to a value of 5.

## DMC Features

The DMC includes a protocol controller that supports JESD79-2E compatible double data rate (DDR2) SDRAM and JESD209A low power DDR (LPDDR) SDRAM devices.

The dynamic memory controller features are listed below.

- Provides 16-bit data interface to SDRAM devices
- Supports a single external rank (one chip select)
- Supports burst lengths of 4 and 8 words
- Provides page hit detection to support multiple column accesses to the same row
- User specified active, precharge and refresh commands.



- Programmable SDRAM access timing parameters
- Enables automatic refresh generation with programmable refresh intervals.
- Self-refresh mode to reduce system power consumption.

The DDR2 features are:

- 256M bit to 2G bit device sizes
- SDRAM data width of 16 (x16 devices only)
- Support for additive latency
- Support for ODT

The LPDDR features are:

- 64M bit to 2G bit device sizes
- SDRAM data width of 16 (x16 devices only)
- Support for deep power down mode
- PHY DLL calibration block
- DDR2 MEMIO I/O buffers
- Efficient transaction processing to improve throughput and bandwidth using:
  - Software programmable SCB IDs to allow SCB ID based priority
  - The ability to postpone up to eight auto-refresh commands
  - Software selectable closed page scheme on a per bank basis
  - Simple transaction scheduling mechanism to reduce read – write turnaround frequency on the memory bus
  - Accesses with the same SCB ID are scheduled back to back to take advantage of same page access in SDRAM
- Create parameters in the SCB slave interface with integrated buffers that support out of order transaction processing
- 

## Feature Exclusions

The DMC exclusions are as follows:

For DDR2:

- 4-bit and 8-bit wide DDR2 DRAM memories are not supported.
- OCD is not supported.
- Burst interleaved accesses are not supported.

For LPDDR:

- 32-bit wide LPDDR memory devices are not supported.
- Status register read (SRR) is not supported
- Sampling the optional Temperature output (TQ) signal is not supported.
- Clock Stop mode is not supported
- Bursts of 2 and 16 words are not supported.
- No support for BURST\_TERMINATE command.
- Dual-die, two CS# and two CKE packages are not supported.

## Functional Description

The Dynamic Memory Controller consists of master and slave interfaces, a protocol controller, and an efficiency controller. These function of these interfaces and controllers are described in the following sections.

### DMC Protocol Controller

The DDR2/LPDDR SDRAM protocol controller translates memory access requests from the SCB (system crossbar) interface to JEDEC protocol specific transactions used by DDR2/LPDDR SDRAM devices.

The protocol controller ensures that the various timing parameters are met before reading and writing the SDRAM. The controller also performs the SDRAM initialization sequence as mandated by the standard. The protocol controller is capable of issuing reads and writes and it can precharge a row in a bank, activate a row in a bank, and also put the SDRAM devices in self refresh and power down.

The protocol controller takes mode register writes from the MMR interface and translates them into mode register writes to SDRAM. Writing into the mode register is restricted via a mask register.

## DMC Efficiency Controller

The efficiency controller controls the ordering of transfers buffered in the read and write command buffers. It attempts to order transfers to optimize the available memory bandwidth. A number of schemes can be used to increase the throughput.

### Read/Write Turnaround

Read/Write turnaround reduces read – write turnaround on the memory bus and is the default method for optimizing bandwidth.

If read and write commands are outstanding in their respective buffers, the state of the efficiency controller determines the direction of the next transfer. When the controller is in the read state, and if both read and write commands are pending, write commands are given priority. Similarly, if the controller is in the write state, and if both read and write commands are pending, read commands are given priority. This ensures that the DMC does not perform transfers in a single direction continuously.

If read commands are outstanding in the read command buffer, a snapshot of the entries is taken. Commands that have the same SCB ID are scheduled in the order in which they were received. When the transfers complete, they are removed from the snapshot. The remaining transfers are scheduled in the order in which they appear in the buffer (0, 1, 2, and 3). After all transactions in the snapshot are complete, the controller moves to the write command buffer.

If write commands are outstanding, a snapshot of all valid entries is taken. A valid write entry is one that has the required amount of data available in the write buffer. Commands are scheduled in the order in which they appear in the buffer (0, 1, 2, and 3). After all transactions in the snapshot are complete they are moved to the read command buffer.

For writes and reads that have a SCB burst that is larger than the DDR2 burst length, one SCB burst is divided into several DDR2 bursts. As soon as the required amount of data is available for a DDR2 burst, a write command is scheduled at the protocol controller. Because of write data interleaving, subsequent SCB data continues to be buffered. If data for the next DDR2 burst is not available for an entry in the snapshot after the first pass is complete, the controller does not wait for the data to become available and moves on to the read snapshot.

For example assume that addresses RD1 and RD2 are present in the read buffer and WR1 and WR2 are present in the write buffer. Each of these addresses has a different SCB ID and the efficiency controller is in the write state.

1. The efficiency controller looks into the read buffer, takes a snapshot of all the commands and prepares to send the commands RD1 and RD2 to the protocol controller.
2. The efficiency controller remembers the location in the buffer where these transactions reside (assume locations 1 and 3). It picks location 1 (RD1) to start transfers at the protocol controller.
3. While this transfer happens, if RD3 arrived at location 2, it is ignored for the current snapshot.

4. After RD1 is finished, the controller moves to location 3 (RD2). Once RD2 completes, the controller moves to the write buffers.
5. While the reads occur, WR3 arrives in the WR address buffer. However, since the interleaving depth is only two, WR1 and WR2 data arrive but WR3's data is still being buffered and the required amount of data is not available. In this case, WR1 and WR2 make it to the snapshot as they are valid and have data ready to be sent. Even though WR3 data may arrive during the transfer of WR1 and WR2, it is only considered in the next snapshot.
6. After WR1 and WR2 are complete, the controller moves to the read buffer.
7. Finally, if WR1 requires two bursts through DDR2 and WR2 requires only one, if all data for WR1 is available, the two DDR2 bursts of WR1 are performed back-to-back before completing WR2. However, if only one DDR2 burst of data is available for WR1, the first DDR2 burst of WR1 is performed, followed by WR2. Then the efficiency controller returns to finish the rest of WR1. If WR1 data is not available, the controller does not wait in the write state and moves to the read buffer.

## Closed Page Per Bank

The DER\_EFFECTL register provides per-bank granularity for closing pages. If software determines that most accesses to a given bank in memory always result in a missed page, the PREC\_BANK bit corresponding to the required bank can be set to close the row after every transfer. This proactive step may result in reduced thrashing and increases memory throughput.

## SCB ID Based Priority

The primary goal of the dynamic memory controller is to improve sustainable memory system bandwidth so that the average request service time can be reduced. However, to service critical requests from any master in the system, a mechanism to elevate priority of a given access is provided. The DMC priority ID register (DMC\_PRI0) can be programmed with up to two SCB IDs whose priority is elevated in the default setup described in [Read/Write Turnaround](#).

After every access in a snapshot, the command buffers are searched to determine whether a command's ID matches with the ID programmed in the DMC\_PRI0 register. If a match occurs and the direction of the access (for example write) is the same as the direction of the snapshot (write), then the priority SCB ID access is sent before sending the subsequent access in the snapshot.

As an alternative to providing priority to a specific SCB ID, if a number of IDs from the same master require priority, the DMC priority mask ID register (DMC\_PRIOMSK) can be programmed so that the corresponding bits are 0. A combination of the DMC\_PRI0 register and the DMC\_PRIOMSK register can then be used to elevate the priority of a select few or all IDs that belong to a master (by default, none of the IDs are prioritized).

The following are a few possibilities

- The `PRI0_ID1_MASK` field of the `DMC_PRIOMSK` register is set to 0000. If a single ID (7234) needs priority, the `PRI0_ID1_MASK` field should be set to FFFF and the `PRI0_ID1` field of the `DMC_PRI0` register is set to 7234
- If the `PRI0_ID1_MASK` field is set to FFFE, the SCB IDs 7234 and 7235 are given priority.
- If the `PRI0_ID1_MASK` field is set to FFFC, the SCB IDs 7234, 7235, 7236 and 7237 are given priority.
- If two transactions with priority are outstanding with one being a read and the other being a write, the priority transaction that does not change the direction of the DMC access gets priority.
- The other priority transaction is handled at the beginning of the next snapshot. For example, if a write snapshot is in progress, the write priority transaction is sent. The read priority transaction is sent at the beginning of the next read snapshot.

**NOTE:** SCB-ID-based priority should be used judiciously because it can significantly reduce the throughput of the DMC.

## Delaying up to Eight Auto-Refresh Commands

This method is used to ensure that auto-refresh does not come in the way of any critical data transfers. Up to eight auto-refresh commands can be accumulated in the DMC and the exact number of auto-refresh commands can be programmed using the `NUM_REF` bit in the DMC efficiency control register.

After the first refresh command is accumulated, the DMC constantly looks for an opportunity to schedule a refresh command. When the SCB read and write command buffers become empty (which implies that no access is outstanding) for the programmed number of clock cycles (`IDLE_CYCLES`) in the DMC efficiency control register, the accumulated number of refresh commands are sent back to back to the DRAM.

After every refresh, the SCB command buffers are checked to ensure they stay empty. However, if the SCB command buffers are always filled, once the programmed number of refresh commands gets accumulated, the refresh operation is elevated to urgent priority and one refresh command is sent immediately. After this, the DMC continues to wait for an opportunity to send out refresh commands. If self-refresh is enabled, all pending refresh commands are given out only after that DMC enters into self-refresh.

## Page Interleaving and Bank Interleaving

In this method, the DMC provides a way to allow consecutive row accesses to fall into the same bank (bank interleaving) or into different bank (page interleaving). The `ADDR_MODE` bit in the `DMC_CTL` register indicates the interleaving option that the DMC is working on. By default, the DMC uses bank interleaving. If the `ADDR_MODE` bit is 1, the DMC uses page interleaving. Page misses in one addressing mode result in hits in the other addressing mode.

## System Crossbar Slave Interface

The system crossbar slave interface is used to move all data. The system crossbar interface accepts interleaved write transactions and is capable of sending out of order response. The read and write interfaces consist of buffers for address, data and control information transferred to/from the system crossbar bus.

The system crossbar interface transactions are sent to the SDRAM only after the SDRAM has been initialized. However if transactions arrive before or during initialization, they will be accumulated in the system crossbar interface and sent out to protocol controller once the initialization completes.

To increase throughput, system crossbar write response is sent out as soon as the final DDR2 burst is scheduled to be transferred into the SDRAM. However, if an auto-refresh needs to be performed, the scheduled write data will be sent only after the auto-refresh. There is a possibility of a delay of a maximum of 64 clock cycles from the moment write response is sent on system crossbar to the actual write of the data into SDRAM.

The system crossbar interface performs the following operations.

- Buffers read/write command requests from system crossbar bus.
- Processes the requests by converting them to protocol controller user interface transfers.
- Sends and receives data to/from the protocol controller.
- Creates suitable read/write response and sends read data back to the system crossbar bus.

The system crossbar slave interface supports the following:

- all burst lengths (1 – 16)
- incremental and wrap bursts
- data transfer sizes of 8, 16 or 32-bits
- arrival of write data before write address
- generation of error responses which includes
  - any access to un-implemented region of the external memory space
  - any access when the SDRAM is in self-refresh, power-down or deep power down (in case of LPDDR)
  - any access when the direct command interface is in operation

## Read/Write Command and Data Buffers

The system crossbar interface comprises of a four deep read command buffer and a four deep write command buffer. Up to four write commands and four read commands can be waiting for access to the SDRAM. The system crossbar write buffer is 32 deep. It can support write data interleaving of two. The system crossbar read buffer is 32 deep.

## Peripheral Bus Slave Interface

The peripheral bus slave interface connects the dynamic memory controller to the peripheral bus and provides a host controller with access to the registers. The peripheral bus slave interface supports the following features:

- read and write word accesses
- 32-bit data bus
- Ability to extend a transfer using PREADY
- Generation of PSLVERR when unimplemented registers are accessed or when read-only registers are written.

## Power Savings

This section discusses the suggested flow to enter and exit DDR self-refresh before and after the processor enters the HIBERNATE state.

For this procedure, the system is in normal operation and the EXT\_WAKE signal is high.

1. Put the SDRAM in self-refresh mode by setting the DMC\_CTL.SRREQ bit.

*STEP RESULT:* The DDR goes through the self-refresh entry sequence and enters the Self Refresh state.

*STEP RESULT:* The DMC\_STAT.SRACK bit is set.

*STEP RESULT:* The CKE pin is driven low by the controller when the DDR has entered Self-Refresh.

2. Read the DMC\_DLLCTL register to get the current DLL tap and calibration settings. This can be used later to quickly lock and start normal DDR operation. The values read are stored in the DPM registers.
3. Initialize the Power-On Reset Delay register to the appropriate values to count off the time required for core  $V_{DD}$  to reach a safe value when exiting the Hibernate state.
4. Enter the Hibernate state by following the procedure detailed in the DPM chapter. Hibernate is indicated by the EXT\_WAKE signal going low.
5. When EXT\_WAKE goes low, the part enters Hibernate state and remains here until brought out by this state through SYS\_PWRGD pin/counter expiry.
6. When a wake-up event occurs, first the EXT\_WAKE signal goes high.

*ADDITIONAL INFORMATION:* When core  $V_{DD}$  power reaches a proper value, the Core domain logic is reset. The DDR controller drives the input of the CKE pad low. When this counter reaches 0 a Counter Expiry signal is generated and Core  $V_{DD}$  is deemed to have reached a safe value. Now, the CKE pin which is driven by the DDR controller is released, so whatever is driven by the DDR controller (at this time driven to 0) at its input pin is driven out to the pads.

7. The DDR controller acts as if it has just come out of reset. The software should now program the `DMC_CTL.SRREQ` bit to write a 1. The `DMC_CTL.INIT` bit should not be set during this write, because starting an `INIT` sequence is not desired.

*ADDITIONAL INFORMATION:* The controller interprets this Self-Refresh request as a command to directly jump to the Self-Refresh state in the state machine.

*ADDITIONAL INFORMATION:* Once the DDR state machine goes to the Self-Refresh state, it should set up all the state variables and status indications to appropriate values. It should appear as though the state machine just entered a Self-Refresh state (as in step 2 above).

*ADDITIONAL INFORMATION:* Further, the `INIT_DONE` bit is now set to 1 (programs should not be performing an `INIT` now because it was already done before Hibernate) and also the `SRACK` bit is set to 1.

8. Software retrieves the DLL data stored in the HV domain and programs the `DMC_DLLCTL` register. This acts as a guess value for the DLL lock and DLL calibration and shortens the time taken to achieve DLL lock.
9. The software should poll for DLL lock status to go high and then clear the `SRREQ` bit in the DDR control register which kick starts a Self-Refresh exit process.
10. After Self-Refresh exit (indicated by the `DMC_STAT.SRACK` bit), start DLL calibration by writing into the `DMC_CTL.DLLCAL` bit. The number of reads required for calibration this time are less than normal power up (cold start) calibration, since the guess value for this has been programmed in the `DMC_DLLCTL` register.

*RESULT:*

Once the calibration process is complete, the `DMC_STAT.DLLCALDONE` bit is set. Normal operations to DDR memory can now start.

## DMC DMA

The DMC controller supports DMA-based transfers to and from external DDR2/LPDDR memory and internal memory.

The DMC DMA controller, part of the Distributed DMA Engines (DDE) that are dispersed through the infrastructure, connects to the system crossbar fabric.

Two DDEs are used for memory-to-memory DMA (MemDMA). One channel is the source channel, and the second, the destination channel.

DMA transfers on the processor can be descriptor-based or register-based. Register-based DMA allows the processor to directly program DMA control registers to initiate a DMA transfer. On completion, the control registers may be automatically updated with their original setup values for continuous transfer, if needed. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. This sort of transfer allows the chaining together of multiple DMA sequences. In



descriptor-based DMA operations, a DMA channel can be programmed to automatically set up and start another DMA transfer after the current sequence completes.

Please refer to the DDE chapter for further details.

## DMC Programming Model

The dynamic memory controller contains five groups of memory mapped registers. These registers are accessed using the MMR access bus and are described below.

- Control and status registers. These registers control the various operation modes of the dynamic memory controller and also provide status.
- Timing parameter registers. The value programmed in these registers depends on the speed grade of DDR2 SDRAM device used.
- Mode register mirror registers - These shadow registers are copies of the mode registers residing in the SDRAM device.
- PHY control and status registers – These registers are used to control the operation of the PHY.
- PAD control registers – These registers are used to control the various aspects of the I/O pads.

The DMC control registers contain sensitive timing parameters and settings for the DDR SDRAM. These registers are programmed with values that are in the operating range of the DDR used.

Writing to reserved fields or writing any reserved values in register bits may cause the dynamic memory controller to function erroneously.

## Configuring the DMC

### *PREREQUISITE:*

After a processor's hardware or software reset, the DMC clocks are enabled. However the DMC must be configured and initialized before any data transfer can take place. Before programming the DMC and executing the power up (initialization) sequence, ensure that the clock to the SDRAM is enabled after the power has stabilized for the proper amount of time (as specified by the SDRAM specification).

1. Check to first ensure that the DMC is idle and not in the midst of any activity.
2. Please the DMC in self-refresh mode.
3. Program the PLL frequency to a new value (if required).

4. Wait the appropriate number of core cycles to ensure that the DLL has locked.
5. Bring the DMC out of self-refresh mode.
6. Program the DMC0\_CFG, DMC0\_CTL, DMC0\_TR0-2, DMC0\_MR, DMC0\_PHY\_CTL1 and DMC0\_PHY\_CTL3 registers to the appropriate values, to set proper SDRAM cycle timing options (eg.,  $t_{RAS}$ ,  $t_{RC}$ ,  $t_{RP}$ ,  $t_{RCD}$ ,  $t_{WR}$ ,  $t_{FAW}$  are some of the parameters).
7. Program the shadow registers DMC0\_EMR1-3, with the required burst length, CAS latency, additive latency and other parameters.
8. Finally, after these registers are programmed, write the `INIT` bit to the DMC control register (0x0004) to begin the power-up initialization sequence.
9. Wait for the SDRAM initialization sequence to complete.
10. Write the DMC\_PADCTL register with values that reflect the type of SDRAM connected to the processor. Specific bit fields within this register determine if the pad is operating in LPDDR mode or DDR2 mode. The drive strengths required at the pads can also be programmed in this register.

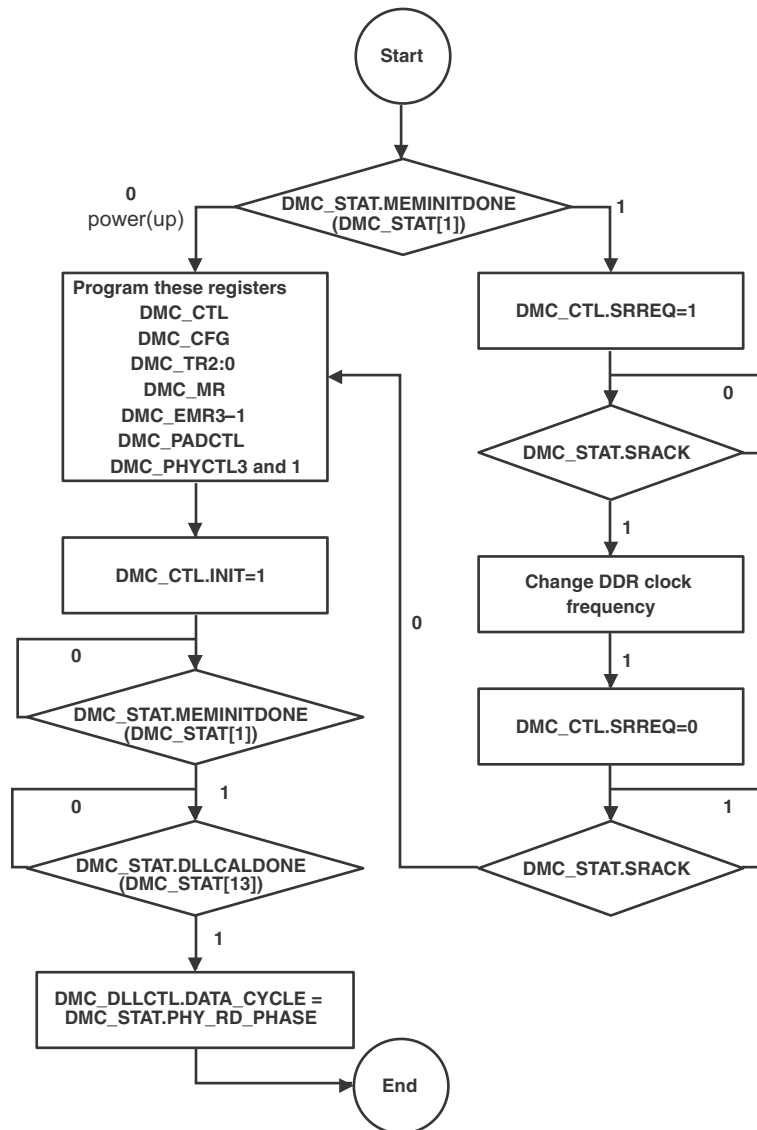


**Figure 11-1:** DMC Initialization Flow

System crossbar transactions that occur during or before initialization are accumulated by the DMC and sent to SDRAM once the SDRAM initialization and/or DLL calibration is complete.

### DLL Initialization

When initializing the DLL for the first time, it is important to follow the following steps:

- 1) Check for (MEMINITDONE & DLLCALDONE) in the DMC\_STAT register
- 2) If this is NOT set, set init bit in DMC\_CTL register and wait for (MEMINITDONE & DLLCALDONE) bits to be set in the DMC\_STAT register

3) If this is set:

- a) Enter into self-refresh
- b) Change the clock frequency, as required
- c) Wait for DLL lock
- d) Exit self refresh
- e) Write new control register values
- f) Set init bit in DMC\_CTL register and poll the MEMINITDONE and DLLCALDONE bits within the DMC\_STAT register
- g) Write to the DMC\_DLLCTL register.

Note that for cases where the DDR2 interface has already been initialized (whether it is via an XML file loaded in during a debug session, or through code executed during the booting process), the user needs to perform second-time initialization as described above.

## DMC Register Descriptions

DDR (DMC) contains the following registers.

**Table 11-2:** DMC Module Registers

| Name        | Description                 |
|-------------|-----------------------------|
| DMC_CTL     | Control Register            |
| DMC_STAT    | Status Register             |
| DMC_EFFCTL  | Efficiency Control Register |
| DMC_PRI0    | Priority ID Register        |
| DMC_PRIOMSK | Priority ID Mask Register   |
| DMC_CFG     | Configuration Register      |
| DMC_TR0     | Timing 0 Register           |
| DMC_TR1     | Timing 1 Register           |

**Table 11-2:** DMC Module Registers (Continued)

| Name         | Description                          |
|--------------|--------------------------------------|
| DMC_TR2      | Timing 2 Register                    |
| DMC_MSK      | Mask (Mode Register Shadow) Register |
| DMC_MR       | Shadow MR Register                   |
| DMC_EMR1     | Shadow EMR1 Register                 |
| DMC_EMR2     | Shadow EMR2 Register                 |
| DMC_EMR3     | Shadow EMR3 Register                 |
| DMC_DLLCTL   | DLL Control Register                 |
| DMC_PHY_CTL1 | PHY Control 1 Register               |
| DMC_PHY_CTL3 | PHY Control 3 Register               |
| DMC_PADCTL   | PAD Control Register                 |

## Control Register

The DMC\_CTL register controls DMC modes, DLL calibration, and DRAM initialization.

**DMC\_CTL: Control Register - R/W**

Reset = 0x0000 0000

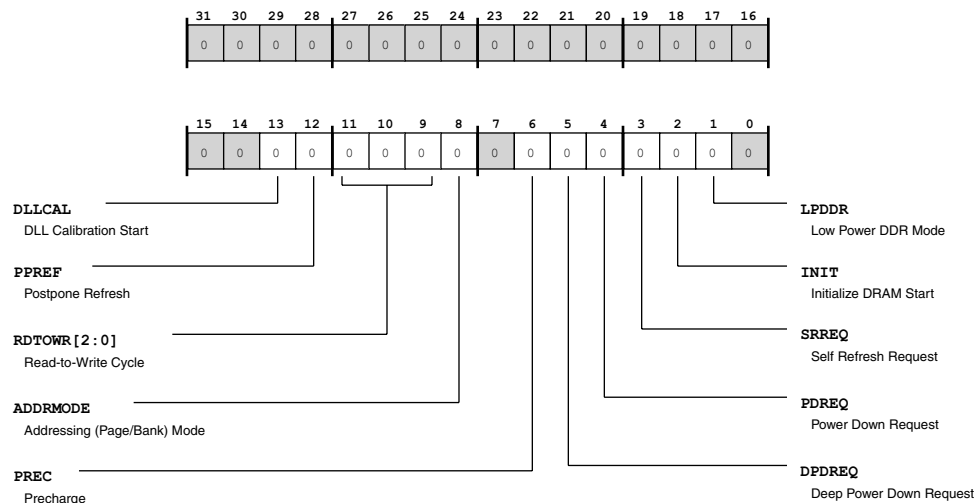


**Figure 11-2: DMC\_CTL Register Diagram**

**Table 11-3: DMC\_CTL Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | DLLCAL   | DLL Calibration Start.<br>The DMC_CTL.DLLCAL bit starts the PHY DLL calibration sequence.<br>Note that this bit always reads as 0.                                                                                                                                                                                                                               |
|         |          | 0 No Effect                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Start PHY DLL Calibration                                                                                                                                                                                                                                                                                                                                      |
| 12      | PPREF    | Postpone Refresh.<br>The DMC_CTL.PPREF bit enables postponing the DMC's sending of auto-refresh commands. When enabled, the DMC accumulates refresh commands. The DMC_EFFECTL.NUMREF field selects the number of refresh commands that the DMC may accumulate. When disabled, the DMC_TR1.TREF field selects the interval for auto-refresh command distribution. |
|         |          | 0 Disable Postpone Refresh                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Enable Postpone Refresh                                                                                                                                                                                                                                                                                                                                        |

**Table 11-3:** DMC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:9    | RDTOWR   | Read-to-Write Cycle.<br>The DMC_CTL.RDTOWR bits select the number of cycles that the DMC adds when a write operation follows a read operation. Note that values 101 through 111 are reserved.                                                                                                                                          |
|         |          | 0      0 Cycles Added                                                                                                                                                                                                                                                                                                                  |
|         |          | 1      1 Cycle Added                                                                                                                                                                                                                                                                                                                   |
|         |          | 2      2 Cycles Added                                                                                                                                                                                                                                                                                                                  |
|         |          | 3      3 Cycles Added                                                                                                                                                                                                                                                                                                                  |
|         |          | 4      4 Cycles Added                                                                                                                                                                                                                                                                                                                  |
| 8       | ADDRMODE | Addressing (Page/Bank) Mode.<br>The DMC_CTL.ADDRMODE bit selects whether the DMC uses page or bank interleaving for addressing. When using page interleaving, the bank address bits follow the most significant column address bits. When using bank interleaving, the bank address bits follow the most significant row address bits. |
|         |          | 0      Bank Interleaving                                                                                                                                                                                                                                                                                                               |
|         |          | 1      Page Interleaving                                                                                                                                                                                                                                                                                                               |
| 6       | PREC     | Precharge.<br>The DMC_CTL.PREC bit enables pre-charge, which closes DRAM rows immediately after access. When disabled, all accesses result in the respective DRAM rows remaining open, until the DMC needs to close them.                                                                                                              |
|         |          | 0      No Effect                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable Precharge                                                                                                                                                                                                                                                                                                                |

**Table 11-3:** DMC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | DPDREQ   | Deep Power Down Request.<br>The DMC_CTL.DPDREQ bit enables deep powerdown mode if low power DMC operation is enabled (DMC_CTL.LPDDR = 1}). When the processor does not require the data stored in SDRAM (assume reset state of SDRAM), the DMC may put the SDRAM in deep powerdown mode. When the DMC is in deep powerdown, any data accesses cause the DMC to generate a bus error. |
|         |          | 0                      Disable Deep Powerdown                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1                      Enable Deep Powerdown                                                                                                                                                                                                                                                                                                                                         |
| 4       | PDREQ    | Power Down Request.<br>The DMC_CTL.PDREQ bit enables powerdown mode. When the DMC is in powerdown, any data accesses cause the DMC to generate a bus error.                                                                                                                                                                                                                          |
|         |          | 0                      Disable Powerdown                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Enable Powerdown                                                                                                                                                                                                                                                                                                                                              |
| 3       | SRREQ    | Self Refresh Request.<br>The DMC_CTL.SRREQ bit enables self refresh mode. When the DMC is in self-refresh, any data accesses cause the DMC to generate a bus error.                                                                                                                                                                                                                  |
|         |          | 0                      Disable Self Refresh                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1                      Enable Self Refresh                                                                                                                                                                                                                                                                                                                                           |
| 2       | INIT     | Initialize DRAM Start.<br>The DMC_CTL.INIT bit starts the power up DRAM initialization sequence and DLL calibration sequence. Note that this bit always reads as 0.                                                                                                                                                                                                                  |
|         |          | 0                      No Effect                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1                      Start DRAM Initialization                                                                                                                                                                                                                                                                                                                                     |



**Table 11-3:** DMC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------|
| 1       | LPDDR    | Low Power DDR Mode.<br>The DMC_CTL.LPDDR bit selects whether the DMC operates in low power DDR mode or DDR2 mode. |
|         |          | 0                      DDR2 mode                                                                                  |
|         |          | 1                      LPDDR mode                                                                                 |

Status Register

The DMC\_STAT register indicates status for modes selected with the DMC\_CTL register and indicates status DMC operations.

**DMC\_STAT: Status Register - R/NW**

Reset = 0x0000 0001

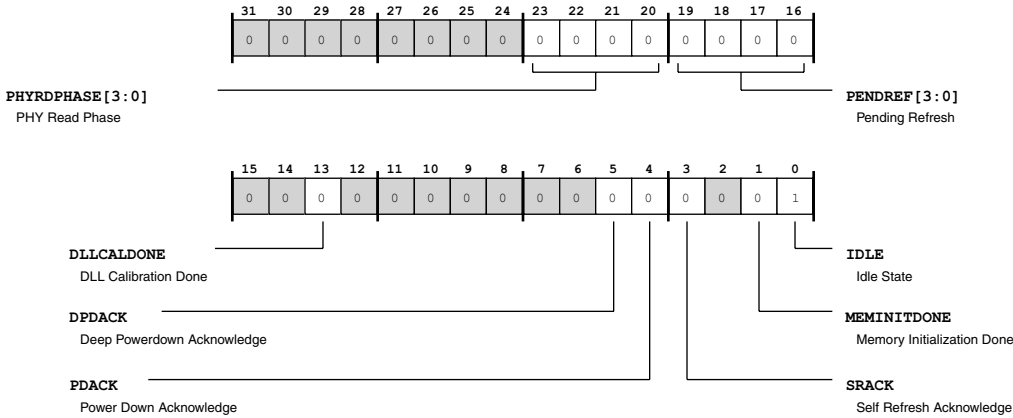


**Figure 11-3:** DMC\_STAT Register Diagram

**Table 11-4:** DMC\_STAT Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                             |
|---------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:20   | PHYRDPHASE | PHY Read Phase.<br>The DMC_STAT.PHYRDPHASE bits indicate the latency after which the DMC may read from the PHY. Taking round trip delay into account, the DLL indicates the exact number of clock cycles after which the controller needs to read data. Values other than those shown are reserved. |
|         |            | 2      2 Clock Cycles Latency                                                                                                                                                                                                                                                                       |
|         |            | 3      3 Clock Cycles Latency                                                                                                                                                                                                                                                                       |
|         |            | 4      4 Clock Cycles Latency                                                                                                                                                                                                                                                                       |
|         |            | 5      5 Clock Cycles Latency                                                                                                                                                                                                                                                                       |
| 19:16   | PENDREF    | Pending Refresh.<br>The DMC_STAT.PENDREF bits indicate the number of pending auto-refresh commands. When the DMC is in low power DDR mode (DMC_CTL.LPDDR=1), the maximum value for DMC_STAT.PENDREF is 3.                                                                                           |
| 13      | DLLCALDONE | DLL Calibration Done.<br>The DMC_STAT.DLLCALDONE indicates that the PHY DLL calibration sequence is complete.                                                                                                                                                                                       |
|         |            | 0      No Status                                                                                                                                                                                                                                                                                    |
|         |            | 1      Completed PHY DLL Calibration                                                                                                                                                                                                                                                                |
| 5       | DPDACK     | Deep Powerdown Acknowledge.<br>The DMC_STAT.DPDACK bit indicates that deep powerdown mode is active. Note that this status is available in low power DDR mode (DMC_CTL.LPDDR=1) only.                                                                                                               |
|         |            | 0      Not in Deep Powerdown Mode                                                                                                                                                                                                                                                                   |
|         |            | 1      Deep Powerdown Mode Active                                                                                                                                                                                                                                                                   |

**Table 11-4:** DMC\_STAT Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                         |
|---------|-------------|-------------------------------------------------------------------------------------------------|
| 4       | PDACK       | Power Down Acknowledge.<br>The DMC_STAT.PDACK bit indicates that powerdown mode is active.      |
|         |             | 0 Not in Powerdown Mode                                                                         |
|         |             | 1 Powerdown Mode Active                                                                         |
| 3       | SRACK       | Self Refresh Acknowledge.<br>The DMC_STAT.SRACK bit indicates that self refresh mode is active. |
|         |             | 0 Not in Self Refresh Mode                                                                      |
|         |             | 1 Self Refresh Mode Active                                                                      |
| 1       | MEMINITDONE | Memory Initialization Done.                                                                     |
|         |             | 0 Init not done                                                                                 |
|         |             | 1 Init complete                                                                                 |
| 0       | IDLE        | Idle State.<br>The DMC_STAT.IDLE bit indicates whether the DMC is idle or busy.                 |
|         |             | 0 Busy                                                                                          |
|         |             | 1 Idle                                                                                          |

## Efficiency Control Register

The DMC\_EFFCTL register control DMC features that improve throughput efficiency. These include features such as auto-refresh management, pre-charge options, and write data options.

DMC\_EFFCTL: Efficiency Control Register - R/W

Reset = 0x0044 0000

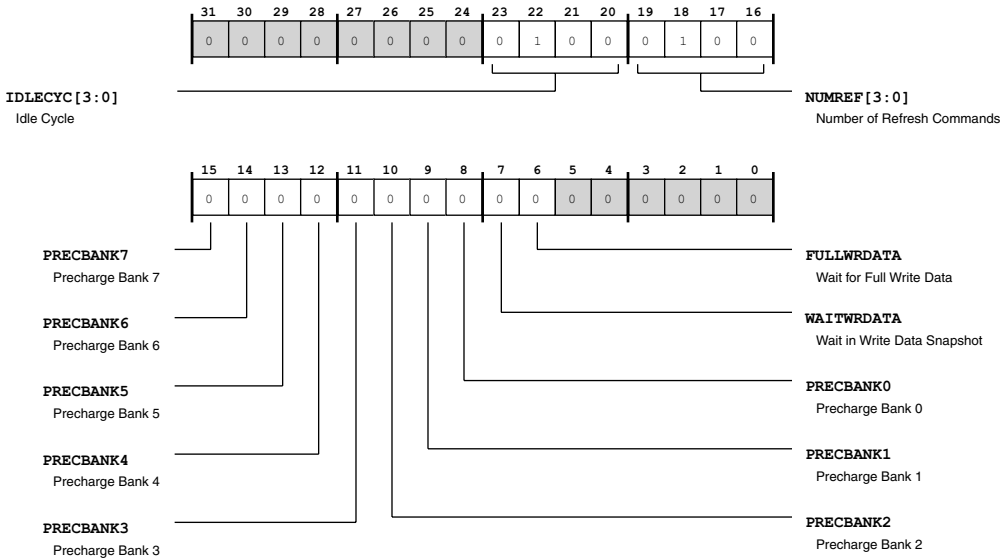


Figure 11-4: DMC\_EFFCTL Register Diagram

Table 11-5: DMC\_EFFCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:20   | IDLECYC  | Idle Cycle.<br>The DMC_EFFCTL.IDLECYC bits select the number of cycles after which the DMC issues any accumulated auto-refresh commands if postpone refresh is enabled (DMC_CTL.PPREF =1). When DMC_EFFCTL.IDLECYC is set to 0, the DMC ignores the DMC_CTL.PPREF selection and does not accumulate/postpone periodic auto refresh commands. |
|         |          | xxxx Idle Cycles to Postpone Refresh Commands                                                                                                                                                                                                                                                                                                |

**Table 11-5:** DMC\_EFFCTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:16   | NUMREF    | <p>Number of Refresh Commands.</p> <p>The <code>DMC_EFFCTL.NUMREF</code> bits select the number of auto-refresh commands that the DMC may accumulate if postpone refresh is enabled (<code>DMC_CTL.PPREF = 1</code>). The number of auto-refresh commands that may be accumulated depends on whether the DMC is in DDR2 or LPDDR mode as selected by the <code>DMC_CTL.LPDDR</code> bit. In LPDDR mode, the DMC may accumulate up to four auto-refresh commands. In DDR2 mode, the DMC may accumulate up to eight auto-refresh commands.</p> |
|         |           | 0 No Refresh Commands Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |           | 1 1 Refresh Command May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |           | 2 2 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 3 3 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 4 4 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 5 5 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 6 6 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 7 7 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 8 8 Refresh Commands May Accumulate                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 15      | PRECBANK7 | <p>Precharge Bank 7.</p> <p>The <code>DMC_EFFCTL.PRECBANK7</code> bit enables precharge (closes the page) of bank 7 after each transfer if the DMC precharge feature is enabled (<code>DMC_CTL.PREC = 1</code>).</p>                                                                                                                                                                                                                                                                                                                         |
|         |           | 0 Disable Precharge Bank 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |           | 1 Enable Precharge Bank 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 11-5:** DMC\_EFFCTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                        |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | PRECBANK6 | Precharge Bank 6.<br>The DMC_EFFCTL.PRECBANK6 bit enables precharge (closes the page) of bank 6 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). |
|         |           | 0      Disable Precharge Bank 6                                                                                                                                                |
|         |           | 1      Enable Precharge Bank 6                                                                                                                                                 |
| 13      | PRECBANK5 | Precharge Bank 5.<br>The DMC_EFFCTL.PRECBANK5 bit enables precharge (closes the page) of bank 5 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). |
|         |           | 0      Disable Precharge Bank 5                                                                                                                                                |
|         |           | 1      Enable Precharge Bank 5                                                                                                                                                 |
| 12      | PRECBANK4 | Precharge Bank 4.<br>The DMC_EFFCTL.PRECBANK4 bit enables precharge (closes the page) of bank 4 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). |
|         |           | 0      Disable Precharge Bank 4                                                                                                                                                |
|         |           | 1      Enable Precharge Bank 4                                                                                                                                                 |
| 11      | PRECBANK3 | Precharge Bank 3.<br>The DMC_EFFCTL.PRECBANK3 bit enables precharge (closes the page) of bank 3 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1). |
|         |           | 0      Disable Precharge Bank 3                                                                                                                                                |
|         |           | 1      Enable Precharge Bank 3                                                                                                                                                 |

**Table 11-5:** DMC\_EFFCTL Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                           |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | PRECBANK2  | Precharge Bank 2.<br>The DMC_EFFCTL.PRECBANK2 bit enables precharge (closes the page) of bank 2 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1)                                                                     |
|         |            | 0      Disable Precharge Bank 2                                                                                                                                                                                                                   |
|         |            | 1      Enable Precharge Bank 2                                                                                                                                                                                                                    |
| 9       | PRECBANK1  | Precharge Bank 1.<br>The DMC_EFFCTL.PRECBANK1 bit enables precharge (closes the page) of bank 1 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1).                                                                    |
|         |            | 0      Disable Precharge Bank 1                                                                                                                                                                                                                   |
|         |            | 1      Enable Precharge Bank 1                                                                                                                                                                                                                    |
| 8       | PRECBANK0  | Precharge Bank 0.<br>The DMC_EFFCTL.PRECBANK0 bit enables precharge (closes the page) of bank 0 after each transfer if the DMC precharge feature is enabled (DMC_CTL.PREC =1).                                                                    |
|         |            | 0      Disable Precharge Bank 0                                                                                                                                                                                                                   |
|         |            | 1      Enable Precharge Bank 0                                                                                                                                                                                                                    |
| 7       | WAITWRDATA | Wait in Write Data Snapshot.<br>The DMC_EFFCTL.WAITWRDATA bit enables waiting in write snapshot if the DDR2 or LPDDR burst is not available for the transfer. If disabled, the DMC does not wait in write snapshot when the burst is unavailable. |
|         |            | 0      Disable Wait for Burst                                                                                                                                                                                                                     |
|         |            | 1      Enable Wait for Burst                                                                                                                                                                                                                      |

Table 11-5: DMC\_EFFCTL Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                               |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | FULLWRDATA | Wait for Full Write Data.<br>The DMC_EFFCTL.FULLWRDATA bit enables waiting until all data is available from the bus to start the DMC transfer. If disabled, the DMC does to wait for full write data, instead the DMC starts the transfer when DDR2 or LPDDR burst becomes available. |
|         |            | 0                      Disable Wait for Full Data                                                                                                                                                                                                                                     |
|         |            | 1                      Enable Wait for Full Data                                                                                                                                                                                                                                      |

Priority ID Register

The DMC\_PRIO register selects up to two internal bus master IDs for banks to receive elevated access priority by the DMC efficiency controller.

DMC\_PRIO: Priority ID Register - R/W

Reset = 0x0000 0000

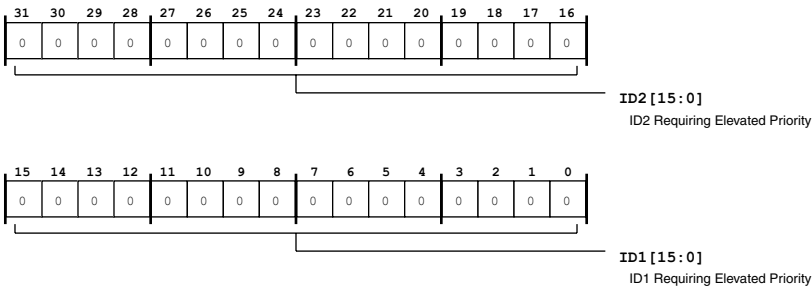


Figure 11-5: DMC\_PRIO Register Diagram

Table 11-6: DMC\_PRIO Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 31:16   | ID2      | ID2 Requiring Elevated Priority. |
| 15:0    | ID1      | ID1 Requiring Elevated Priority. |



## Priority ID Mask Register

The DMC\_PRIOMSK register permits masking portions of up to two internal bus master IDs for banks to receive elevated access priority by the DMC efficiency controller. This masking provides for elevating the access priority of either a single ID or a range of IDs.

DMC\_PRIOMSK: Priority ID Mask Register - R/W

Reset = 0x0000 0000

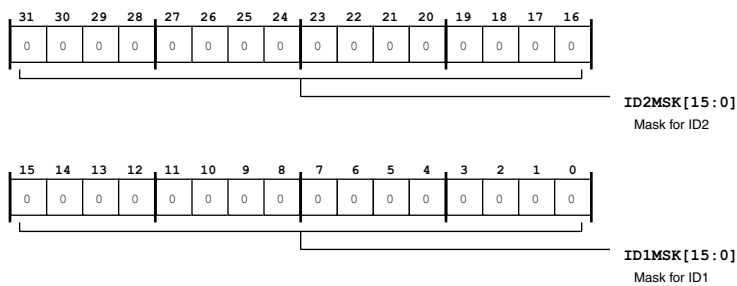


Figure 11-6: DMC\_PRIOMSK Register Diagram

Table 11-7: DMC\_PRIOMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | ID2MSK   | Mask for ID2.           |
| 15:0    | ID1MSK   | Mask for ID1.           |

## Configuration Register

The DMC\_CFG register selects SDRAM device specific parameters and selects the SDRAM interface width.

DMC\_CFG: Configuration Register - R/W

Reset = 0x0000 0000

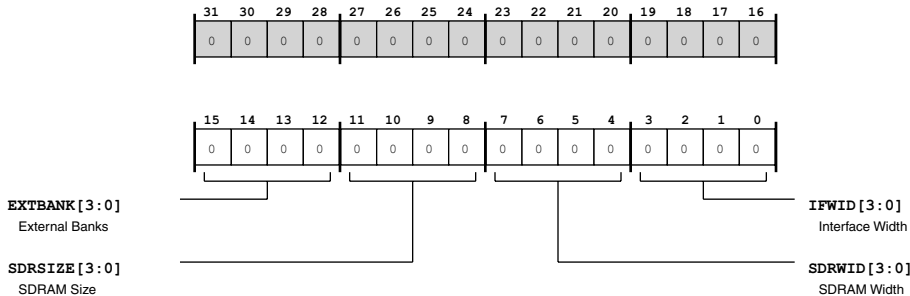


Figure 11-7: DMC\_CFG Register Diagram

Table 11-8: DMC\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 15:12   | EXTBANK  | External Banks.<br>The DMC_CFG.EXTBANK bits select the number of external banks connected to the DMC. Note that all values other than those shown are reserved. |                             |
|         |          | 0                                                                                                                                                               | 1 External Bank             |
| 11:8    | SDRSIZE  | SDRAM Size.<br>The DMC_CFG.SDRSIZE bits select the size of individual SDRAM connected to the DMC. Note that all values other than those shown are reserved.     |                             |
|         |          | 0                                                                                                                                                               | 64M Bit SDRAM (LPDDR Only)  |
|         |          | 1                                                                                                                                                               | 128M Bit SDRAM (LPDDR Only) |
|         |          | 2                                                                                                                                                               | 256M Bit SDRAM              |
|         |          | 3                                                                                                                                                               | 512M Bit SDRAM              |
|         |          | 4                                                                                                                                                               | 1G Bit SDRAM                |
|         |          | 5                                                                                                                                                               | 2G Bit SDRAM                |

**Table 11-8:** DMC\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | SDRWID   | SDRAM Width.<br>The DMC_CFG.SDRWID bits select the width of the individual SDRAM connected to the DMC. Note that all values other than those shown are reserved.  |
|         |          | 2      16-Bit Wide SDRAM                                                                                                                                          |
| 3:0     | IFWID    | Interface Width.<br>The DMC_CFG.IFWID bits select the width of the interface between the DMC and SDRAM. Note that all values other than those shown are reserved. |
|         |          | 2      16-Bit Wide Interface<br>All Other Values: Reserved This field specifies the interface width between the controller and the SDRAM.                         |

## Timing 0 Register

The DMC\_TR0 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and must be programmed before initializing the SDRAM. Note that all values for bit fields in DMC\_TR0 are in increments of clock cycle time ( $t_{CK}$ ).

### DMC\_TR0: Timing 0 Register - R/W

Reset = 0x0000 0000

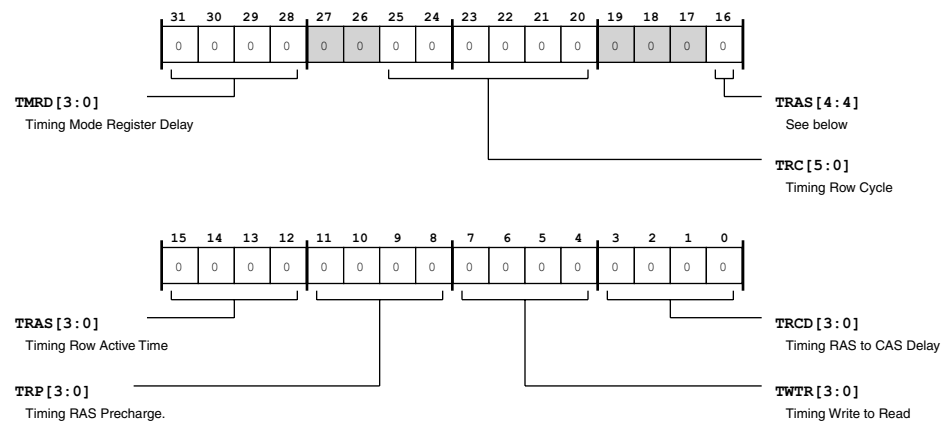


**Figure 11-8:** DMC\_TR0 Register Diagram

**Table 11-9:** DMC\_TR0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:28   | TMRD     | Timing Mode Register Delay.<br>The DMC_TR0.TMRD field selects the set-to-active timing parameter ( $t_{MRD}$ ), which is the number of clock cycles that occur after the mode registers in the SDRAM are set and before the next command is issued.      |
| 25:20   | TRC      | Timing Row Cycle.<br>The DMC_TR0.TRC field selects the active-to-active time ( $t_{RC}$ ), which is the minimum number of clock cycles that occur from an active command to the next active command in the same bank.                                    |
| 16:12   | TRAS     | Timing Row Active Time.<br>The DMC_TR0.TRAS field selects the active-to-precharge time ( $t_{RAS}$ ), which is the number of clock cycles that occur from an active command until a precharge command is allowed.                                        |
| 11:8    | TRP      | Timing RAS Precharge..<br>The DMC_TR0.TRP field selects the precharge-to-active time ( $t_{RP}$ ), which is the number of clock cycles that occur while the SDRAM recovers from a precharge command and becomes ready to accept the next active command. |
| 7:4     | TWTR     | Timing Write to Read.<br>The DMC_TR0.TWTR field selects the write-to-read delay time ( $t_{WTR}$ ), which is the number of clock cycles that occur from the last write data to the next read command.                                                    |
| 3:0     | TRCD     | Timing RAS to CAS Delay.<br>The DMC_TR0.TRCD field selects the RAS to CAS delay time ( $t_{RCD}$ ), which is the number of clock cycles that occur from an active command to a read/write assertion.                                                     |

## Timing 1 Register

The DMC\_TR1 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and must be programmed before initializing the SDRAM. Note that all values for bit fields in DMC\_TR1 are in increments of clock cycle time ( $t_{CK}$ ).

**DMC\_TR1: Timing 1 Register - R/W**

Reset = 0x0000 0000

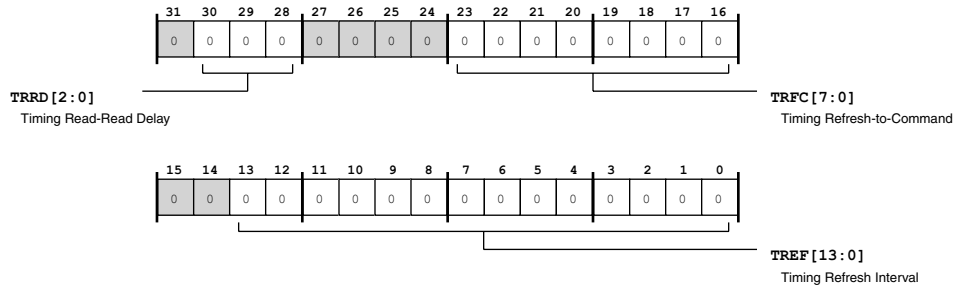


**Figure 11-9:** DMC\_TR1 Register Diagram

**Table 11-10:** DMC\_TR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30:28   | TRRD     | Timing Read-Read Delay.<br>The DMC_TR1 . TRRD field selects the active-to-active time ( $t_{RRD}$ ), which is the minimum number of clock cycles occurring from a bank x active command to a bank y active command.                                                                                                                                                                                                                                                                                                                                            |
| 23:16   | TRFC     | Timing Refresh-to-Command.<br>The DMC_TR1 . TRFC field selects the refresh-to-active command delay ( $t_{RFC}$ ), which is the number of clock cycles required for the SDRAM to recover from a refresh signal to be ready to take the next command. It is also the number of clock cycles needed for the SDRAM to recover from executing one active command and ready to accept the next active command.                                                                                                                                                       |
| 13:0    | TREF     | Timing Refresh Interval.<br>The DMC_TR1 . TREF field selects the refresh interval time ( $t_{REF}$ ), which is the number of clock cycles occurring from one refresh command to the next refresh command. The actual timing of issuing a precharge command may be delayed by if the SDRAM is processing a normal access. However, the delay is not accumulative so there is no need to shorten the refresh interval to account for the memory access time. The non-accumulative refresh delay typically increases memory bandwidth by a few percentage points. |

## Timing 2 Register

The DMC\_TR2 register selects timing parameters for DMC operation to corresponding with parameters of the SDRAM device that is used in the system. The timing registers must be programmed to match the device for correct operation of the SDRAM and must be programmed before initializing the SDRAM. Note that all values for bit fields in DMC\_TR2 are in increments of clock cycle time ( $t_{CK}$ ).

**DMC\_TR2: Timing 2 Register - R/W**

Reset = 0x0000 0000

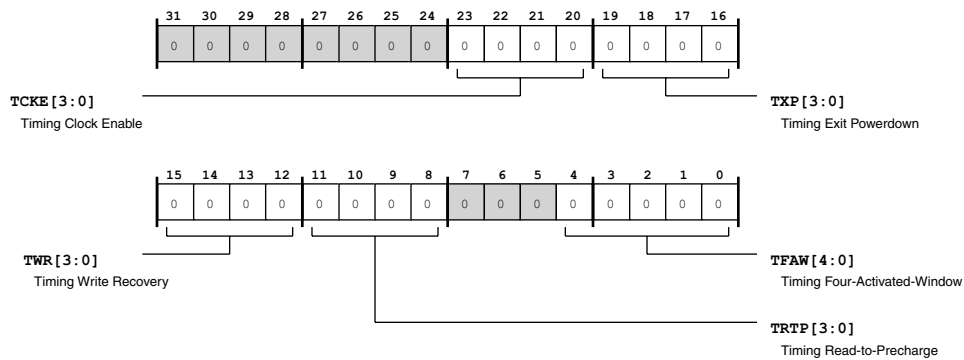


**Figure 11-10:** DMC\_TR2 Register Diagram

**Table 11-11:** DMC\_TR2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 23:20   | TCKE     | Timing Clock Enable.<br>The DMC_TR2.TCKE field selects the CKE minimum pulsewidth ( $t_{CKE}$ ).                                              |
| 19:16   | TXP      | Timing Exit Powerdown.<br>The DMC_TR2.TXP field selects the exit powerdown to next valid command time ( $t_{XP}$ ).                           |
| 15:12   | TWR      | Timing Write Recovery.<br>The DMC_TR2.TWR field selects the write recovery time ( $t_{WR}$ ). Note that this parameter applies to LPDDR only. |
| 11:8    | TRTP     | Timing Read-to-Precharge.<br>The DMC_TR2.TRTP field selects the internal read to precharge time ( $t_{RTP}$ ).                                |

Table 11-11: DMC\_TR2 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4:0     | TFAW     | Timing Four-Activated-Window.<br>The DMC_TR2.TFAW field selects the four-banks-activated window time ( $t_{FAW}$ ). No more than four SDRAM banks should be activated within this window. |

Mask (Mode Register Shadow) Register

The DMC\_MSK register permits masking (disabling) writes to the MR and EMRn registers in the SDRAM. When masked, writes to these registers go instead to shadow copies of these registers (DMC\_MR, DMC\_EMR1, DMC\_EMR2, and DMC\_EMR3), which are maintained within the DMC. When a shadow register's corresponding bit is unmasked (enabled), the DMC generates the MRS or EMRS command to transfer the contents of the shadow register (in the DMC) to the actual register (in the SDRAM).

DMC\_MSK: Mask (Mode Register Shadow) Register - R/W

Reset = 0x0000 0000

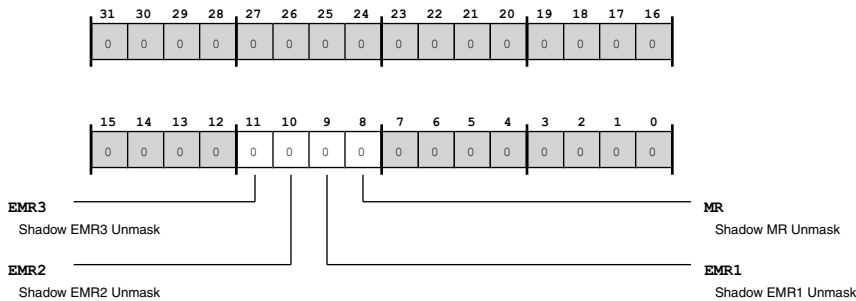


Figure 11-11: DMC\_MSK Register Diagram

**Table 11-12:** DMC\_MSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | EMR3     | Shadow EMR3 Unmask.<br>The DMC_MSK.EMR3 bit masks or unmasks writes to the EMR3 register (in DDR2) in the SDRAM. When masked, writes to this register instead go to the DMC_EMR3 register. When unmasked, the DMC writes the DMC_EMR3 value to the EMR3 register (in DDR2) in the SDRAM. After completing the write, the DMC clears this bit. Note that this bit must not be enabled when in LPDDR mode (DMC_CTL.LPDDR =1). |
|         |          | 0 Mask (Disable) Write to EMR3                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Unmask (Enable) Write to EMR3                                                                                                                                                                                                                                                                                                                                                                                             |
| 10      | EMR2     | Shadow EMR2 Unmask.<br>The DMC_MSK.EMR2 bit masks or unmasks writes to the EMR2 register (in DDR2) or the EMR register (in LPDDR) in the SDRAM. When masked, writes to this register instead go to the DMC_EMR2 register. When unmasked, the DMC writes the DMC_EMR2 value to the EMR2 register (in DDR2) or the EMR register (in LPDDR) in the SDRAM. After completing the write, the DMC clears this bit.                 |
|         |          | 0 Mask (Disable) Write to EMR2                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Unmask (Enable) Write to EMR2                                                                                                                                                                                                                                                                                                                                                                                             |
| 9       | EMR1     | Shadow EMR1 Unmask.<br>The DMC_MSK.EMR1 bit masks or unmasks writes to the EMR1 register in the SDRAM. When masked, writes to this register instead go to the DMC_EMR1 register. When unmasked, the DMC writes the DMC_EMR1 value to the EMR1 register in the SDRAM. After completing the write, the DMC clears this bit. Note that this bit must not be enabled when in LPDDR mode (DMC_CTL.LPDDR =1).                     |
|         |          | 0 Mask (Disable) Write to EMR1                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Unmask (Enable) Write to EMR1                                                                                                                                                                                                                                                                                                                                                                                             |



Table 11-12: DMC\_MSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | MR       | Shadow MR Unmask.<br>The DMC_MSK.MR bit masks or unmasks writes to the MR register in the SDRAM. When masked, writes to this register instead go to the DMC_MR register. When unmasked, the DMC writes the DMC_MR value to the MR register in the SDRAM. After completing the write, the DMC clears this bit. |
|         |          | 0 Mask (Disable) Write to MR                                                                                                                                                                                                                                                                                  |
|         |          | 1 Unmask (Enable) Write to MR                                                                                                                                                                                                                                                                                 |

Shadow MR Register

The DMC\_MR register in the DMC shadows the MR register in the SDRAM when the DMC is in DDR2 mode or LPDDR mode (DMC\_CTL.LPDDR = 0 or = 1). If unmasked by the corresponding bit in the shadow mask register (DMC\_MSK.MR = 1), a write to DMC\_MR triggers a "mode register set" command on the memory interface. If masked, a write to DMC\_MR only updates the register in the DMC, not the register in the SDRAM.

DMC\_MR: Shadow MR Register - R/W

Reset = 0x0000 0000

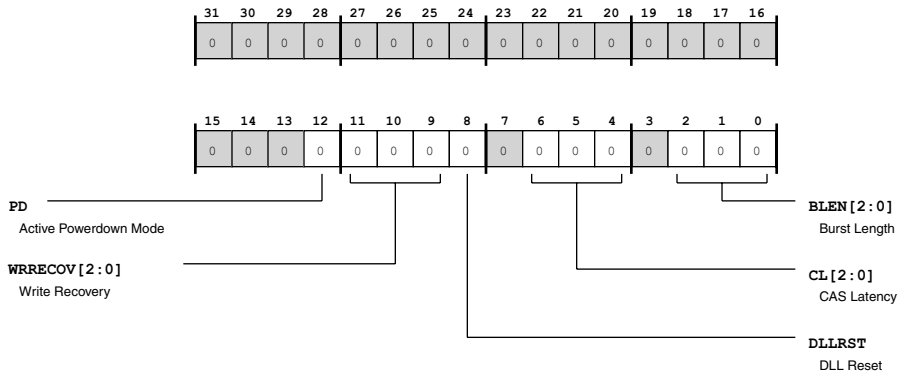


Figure 11-12: DMC\_MR Register Diagram

**Table 11-13:** DMC\_MR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | PD       | Active Powerdown Mode.<br>The <code>DMC_MR.PD</code> bit selects the active powerdown mode. Note that this parameter applies only for DDR2 mode and is reserved for LPDDR mode. For more information about this mode, see the data sheet for the SDRAM being used in your system.                                   |
|         |          | 0 Fast exit (normal)                                                                                                                                                                                                                                                                                                |
|         |          | 1 Slow exit (low power)                                                                                                                                                                                                                                                                                             |
| 11:9    | WRRECOV  | Write Recovery.<br>The <code>DMC_MR.WRRECOV</code> bit selects the write recovery time in terms of clock cycles ( $t_{CK}$ ). Note that this parameter applies only for DDR2 mode and is reserved for LPDDR mode. For more information about this mode, see the data sheet for the SDRAM being used in your system. |
|         |          | 1 2 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 2 3 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 3 4 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 4 5 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 5 6 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 6 7 Clock Cycles                                                                                                                                                                                                                                                                                                    |
|         |          | 7 8 Clock Cycles                                                                                                                                                                                                                                                                                                    |
| 8       | DLLRST   | DLL Reset.<br>The <code>DMC_MR.DLLRST</code> bit initiates a DLL reset on the SDRAM. Note that this parameter applies only for DDR2 mode and is reserved for LPDDR mode. For more information about this operation, see the data sheet for the SDRAM being used in your system.                                     |
|         |          | 0 Normal Operation                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Reset DLL                                                                                                                                                                                                                                                                                                         |

**Table 11-13:** DMC\_MR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:4     | CL       | CAS Latency.<br>The <code>DMC_MR.CL</code> bits select latency from the assertion of a read/write signal to the SDRAM until the first valid data on the output from the SDRAM in terms of clock cycles ( $t_{CK}$ ). For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 2      2 clock cycle latency                                                                                                                                                                                                                                                                                                |
|         |          | 3      3 clock cycle latency                                                                                                                                                                                                                                                                                                |
|         |          | 4      4 clock cycle latency (DDR2)                                                                                                                                                                                                                                                                                         |
|         |          | 5      5 clock cycle latency (DDR2)                                                                                                                                                                                                                                                                                         |
|         |          | 6      6 clock cycle latency (DDR2)                                                                                                                                                                                                                                                                                         |
| 2:0     | BLEN     | Burst Length.<br>The <code>DMC_MR.BLEN</code> bits select burst length for transfers. For more information about this operation, see the data sheet for the SDRAM being used in your system. Note that values other than those shown are not supported.                                                                     |
|         |          | 2      4-Bit Burst Length                                                                                                                                                                                                                                                                                                   |
|         |          | 3      8-Bit Burst Length                                                                                                                                                                                                                                                                                                   |

## Shadow EMR1 Register

The `DMC_EMR1` register in the DMC shadows the EMR1 register in the SDRAM when the DMC is in DDR2 mode (`DMC_CTL.LPDDR=0`). Note that this register must not be used when the DMC is in LPDDR mode (`DMC_CTL.LPDDR=1`). If unmasked by the corresponding bit in the shadow mask register (`DMC_MSK.EMR1=1`), a write to `DMC_EMR1` triggers an extended "mode register set" command on the memory interface. If masked, a write to `DMC_EMR1` only updates the register in the DMC, not the register in the SDRAM.

DMC\_EMR1: Shadow EMR1 Register - R/W

Reset = 0x0000 0000

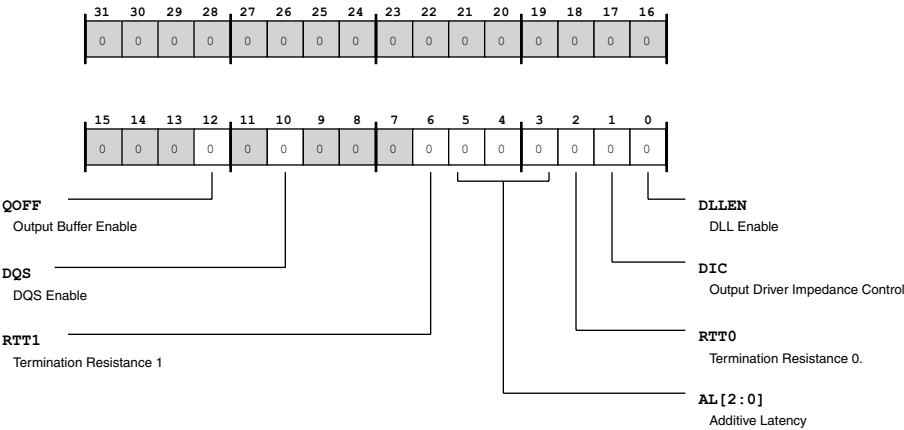


Figure 11-13: DMC\_EMR1 Register Diagram

Table 11-14: DMC\_EMR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | QOFF     | Output Buffer Enable.<br>The DMC_EMR1 . QOFF bit enables the SDRAM output pins. For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 0      Enable                                                                                                                                                                          |
|         |          | 1      Disable                                                                                                                                                                         |
| 10      | DQS      | DQS Enable.<br>The DMC_EMR1 . DQS bit enables operation of the DQS pin. For more information about this operation, see the data sheet for the SDRAM being used in your system.         |
|         |          | 0      Enable                                                                                                                                                                          |
|         |          | 1      Disable                                                                                                                                                                         |

**Table 11-14:** DMC\_EMR1 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | RTT1     | Termination Resistance 1.<br>The DMC_EMR1 . RTT1 bit combines with the DMC_EMR1 . RTT0 bit to set the termination resistance. See the DMC_EMR1 . RTT0 bit description for more information.                                                                                                                                                                                                                                        |
|         |          | 0                      Disable RTT1                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1                      Enable RTT1                                                                                                                                                                                                                                                                                                                                                                                                 |
| 5:3     | AL       | Additive Latency.<br>The DMC_EMR1 . AL bits select a number of added latency time for CAS operations in terms of clock cycles ( $t_{CK}$ ). For more information about this operation, see the data sheet for the SDRAM being used in your system.                                                                                                                                                                                 |
|         |          | 0                      0 Clock Cycles Added                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1                      1 Clock Cycle Added                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 2                      2 Clock Cycles Added                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 3                      3 Clock Cycles Added                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 4                      4 Clock Cycles Added                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 5                      5 Clock Cycles Added                                                                                                                                                                                                                                                                                                                                                                                        |
| 2       | RTT0     | Termination Resistance 0..<br>The DMC_EMR1 . RTT0 bit and the DMC_EMR1 . RTT1 bits select the SDRAM termination resistance.<br>RTT1=0, RTT0=0 : No ODT at memory device<br>RTT1=0, RTT0=1 : 75 Ohm ODT at memory device<br>RTT1=1, RTT0=0 : 150 Ohm ODT at memory device<br>RTT1=1, RTT0=1 : 50 Ohm ODT at memory device<br>For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 0                      Disable RTT0                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1                      Enable RTT0                                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 11-14:** DMC\_EMR1 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | DIC      | Output Driver Impedance Control.<br>The DMC_EMR1.DIC bit selects the drive strength mode for the SDRAM. For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 0 Full Strength                                                                                                                                                                                                |
|         |          | 1 Reduced Strength                                                                                                                                                                                             |
| 0       | DLLEN    | DLL Enable.<br>The DMC_EMR1.DLLEN bit enables the DLL in the SDRAM. For more information about this operation, see the data sheet for the SDRAM being used in your system.                                     |
|         |          | 0 Enable DLL (Normal Operation)                                                                                                                                                                                |
|         |          | 1 Disable DLL (Test/Debug Operation)                                                                                                                                                                           |

## Shadow EMR2 Register

The DMC\_EMR2 register in the DMC shadows the EMR2 register in the SDRAM when the DMC is in DDR2 mode (DMC\_CTL.LPDDR=0) and shadows the EMR register in the SDRAM when the DMC is in LPDDR mode (DMC\_CTL.LPDDR=1). If unmasked by the corresponding bit in the shadow mask register (DMC\_MSK.EMR2=1), a write to DMC\_EMR2 triggers an extended "mode register set" command on the memory interface. If masked, a write to DMC\_EMR2 only updates the register in the DMC, not the register in the SDRAM.

### DMC\_EMR2: Shadow EMR2 Register - R/W

Reset = 0x0000 0000

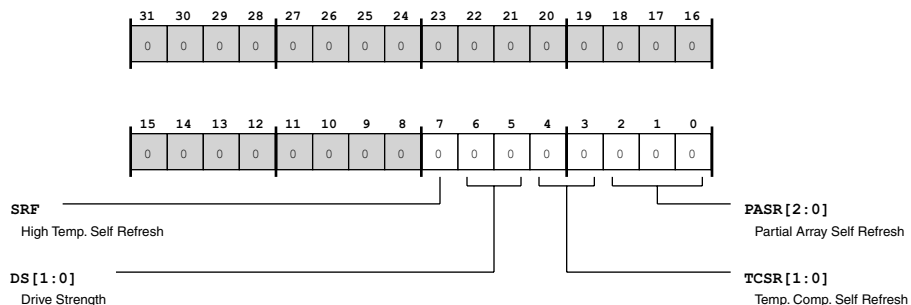


**Figure 11-14:** DMC\_EMR2 Register Diagram

**Table 11-15:** DMC\_EMR2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | SRF      | High Temp. Self Refresh.<br>The DMC_EMR2.SRF bit enables the SDRAM's high temperature self refresh rate feature when the DMC is in DDR2 mode. (This bit is reserved in LPDDR mode.) For more information about this operation, see the data sheet for the SDRAM being used in your system.                                      |
|         |          | 0      1x Refresh Rate (0degC to 85degC)                                                                                                                                                                                                                                                                                        |
|         |          | 1      2x Refresh Rate (>85degC)                                                                                                                                                                                                                                                                                                |
| 6:5     | DS       | Drive Strength.<br>The DMC_EMR2.DS bits select the drive strength value when the DMC is in LPDDR mode. (These bits are reserved when the DMC is in DDR2 mode.) Note that all values other than those shown are reserved. For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 0      Full Drive Strength                                                                                                                                                                                                                                                                                                      |
|         |          | 1      1/2 Drive Strength                                                                                                                                                                                                                                                                                                       |
|         |          | 2      1/4 Drive Strength                                                                                                                                                                                                                                                                                                       |
|         |          | 3      3/4 Drive Strength                                                                                                                                                                                                                                                                                                       |
| 4:3     | TCSR     | Temp. Comp. Self Refresh.<br>The DMC_EMR2.TCSR bits select the temperature for applying temperature compensated self refresh when the DMC is in LPDDR mode. (These bits are reserved when the DMC is in DDR2 mode.) For more information about this operation, see the data sheet for the SDRAM being used in your system.      |
|         |          | 0      70 degree C (in LPDDR Mode)                                                                                                                                                                                                                                                                                              |
|         |          | 1      45 degree C                                                                                                                                                                                                                                                                                                              |
|         |          | 2      15 degree C                                                                                                                                                                                                                                                                                                              |
|         |          | 3      85 degree C                                                                                                                                                                                                                                                                                                              |

**Table 11-15:** DMC\_EMR2 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2:0     | PASR     | Partial Array Self Refresh.<br>The DMC_EMR2.PASR bits select the amount of memory to be refreshed during self refresh. For more information about this operation, see the data sheet for the SDRAM being used in your system. |
|         |          | 0 Full Array (in LPDDR Mode)                                                                                                                                                                                                  |
|         |          | 1 1/2 Array                                                                                                                                                                                                                   |
|         |          | 2 1/4 Array                                                                                                                                                                                                                   |
|         |          | 3 Reserved                                                                                                                                                                                                                    |
|         |          | 4 Reserved                                                                                                                                                                                                                    |
|         |          | 5 1/8 Array                                                                                                                                                                                                                   |
|         |          | 6 1/16 Array                                                                                                                                                                                                                  |
|         |          | 7 Reserved                                                                                                                                                                                                                    |

## Shadow EMR3 Register

The DMC\_EMR3 register in the DMC shadows the EMR3 register in the SDRAM when the DMC is in DDR2 mode (DMC\_CTL.LPDDR=0). Note that this register must not be used when the DMC is in LPDDR mode (DMC\_CTL.LPDDR=1). If unmasked by the corresponding bit in the shadow mask register (DMC\_MSK.EMR3=1), a write to DMC\_EMR3 triggers an extended "mode register set" command on the memory interface. If masked, a write to DMC\_EMR3 only updates the register in the DMC, not the register in the SDRAM.

### DMC\_EMR3: Shadow EMR3 Register - R/W

Reset = 0x0000 0000

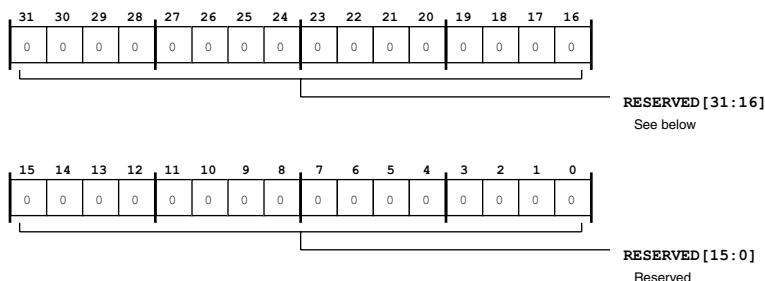


**Figure 11-15:** DMC\_EMR3 Register Diagram



**Table 11-16:** DMC\_EMR3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                   |
|---------|----------|-----------------------------------------------------------|
| 31:0    | RESERVED | Reserved.<br>All bits the DMC_EMR3 register are reserved. |

## DLL Control Register

The DMC\_DLLCTL register holds the programmable parameters associated with the DLLs within the DMC PHY.

**DMC\_DLLCTL: DLL Control Register - R/W**

Reset = 0x0000 034b

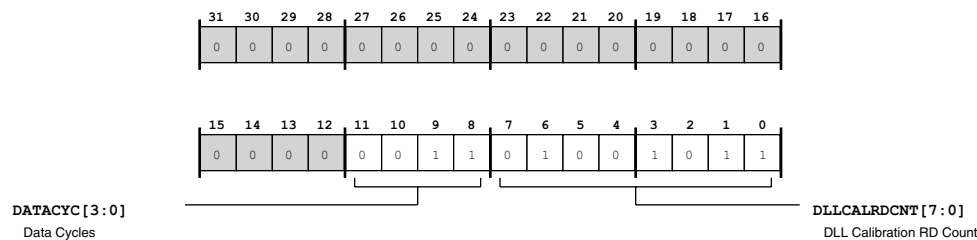


**Figure 11-16:** DMC\_DLLCTL Register Diagram

**Table 11-17:** DMC\_DLLCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:8    | DATA CYC | <p>Data Cycles.</p> <p>The DMC_DLLCTL.DATA CYC bits select the latency after which the DMC reads data from the PHY. This field must be written with the value indicated in the DMC_STAT.PHYRDPHASE field, or data corruption occurs on all SDRAM reads.</p> <p>Taking round trip delay into account, the DLL indicates whether a latency of 2 cycles is supported by means of status bits.</p> |
|         | 2        | 2 Clock Cycles Latency                                                                                                                                                                                                                                                                                                                                                                         |
|         | 3        | 3 Clock Cycles Latency                                                                                                                                                                                                                                                                                                                                                                         |
|         | 4        | 4 Clock Cycles Latency                                                                                                                                                                                                                                                                                                                                                                         |
|         | 5        | 5 Clock Cycles Latency                                                                                                                                                                                                                                                                                                                                                                         |

Table 11-17: DMC\_DLLCTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                    |
|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | DLLCALRDCNT | DLL Calibration RD Count.<br>The DMC_DLLCTL.DLLCALRDCNT field selects the number of read operations that the PHY uses for DLL calibration. |

PHY Control 1 Register

The DMC\_PHY\_CTL1 register controls programmable PHY features.

DMC\_PHY\_CTL1: PHY Control 1 Register - R/W

Reset = 0x0000 0000

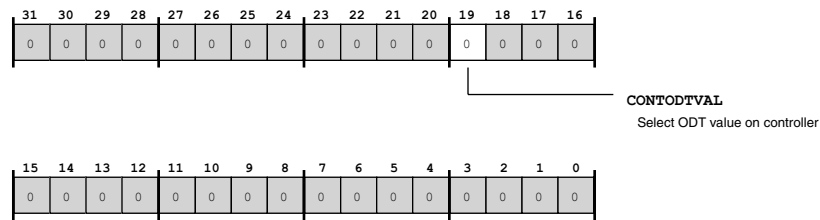


Figure 11-17: DMC\_PHY\_CTL1 Register Diagram

Table 11-18: DMC\_PHY\_CTL1 Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                              |
|---------|------------|----------------------------------------------------------------------------------------------------------------------|
| 19      | CONTODTVAL | Select ODT value on controller.<br>The DMC_PHY_CTL1.CONTODTVAL bit selects the output drive termination (ODT) value. |
|         |            | 0      75 Ohms Termination                                                                                           |
|         |            | 1      150 Ohms Termination                                                                                          |

PHY Control 3 Register

The DMC\_PHY\_CTL3 register controls programmable PHY features.

**DMC\_PHY\_CTL3: PHY Control 3 Register - R/W**

Reset = 0x0000 0000

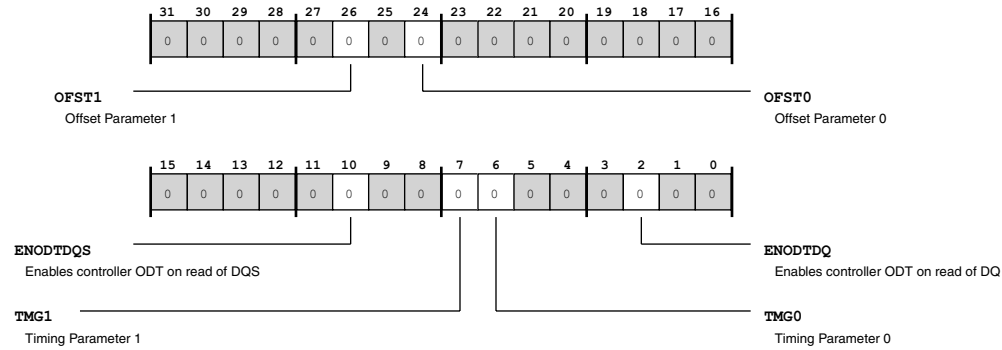


**Figure 11-18:** DMC\_PHY\_CTL3 Register Diagram

**Table 11-19:** DMC\_PHY\_CTL3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | OFST1    | Offset Parameter 1.<br>Controls latching of data by the controller. . See HW Reference for proper setting. Needs to be set to 1 for BF609.                                                                                     |
| 24      | OFST0    | Offset Parameter 0.<br>Controls latching of data by the controller. See section on product specific register and bit settings for more information. (For example, this bit needs to =1 for the ADSP-BF609 Blackfin processor.) |
| 10      | ENODTDQS | Enables controller ODT on read of DQS.<br>The DMC_PHY_CTL3.ENODTDQS bit enables ODT for DQS pads.                                                                                                                              |
|         |          | 0      Disable                                                                                                                                                                                                                 |
|         |          | 1      Enable                                                                                                                                                                                                                  |
| 7       | TMG1     | Timing Parameter 1.<br>Controls latching of data by the controller. Needs to be set to 1 for BF609.                                                                                                                            |
| 6       | TMG0     | Timing Parameter 0.<br>Controls latching of data by the controller. Needs to be set to 1 for BF609.                                                                                                                            |

**Table 11-19:** DMC\_PHY\_CTL3 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                        |         |
|---------|----------|------------------------------------------------------------------------------------------------|---------|
| 2       | ENODTDQ  | Enables controller ODT on read of DQ.<br>The DMC_PHY_CTL3.ENODTDQ bit enables ODT for DQ pads. |         |
|         |          | 0                                                                                              | Disable |
|         |          | 1                                                                                              | Enable  |

**PAD Control Register**

This register allows programming control parameters associated with the DQ and DQS pads of the SDRAM memory interface. This register also allows programming control parameters associated with CK, CKE and CMD pads (RAS\_b, CAS\_b, WE\_b, CS\_b, ODT, CS\_b, A and BA) of the SDRAM memory interface.

**DMC\_PADCTL: PAD Control Register - R/W**

Reset = 0x0004 4400

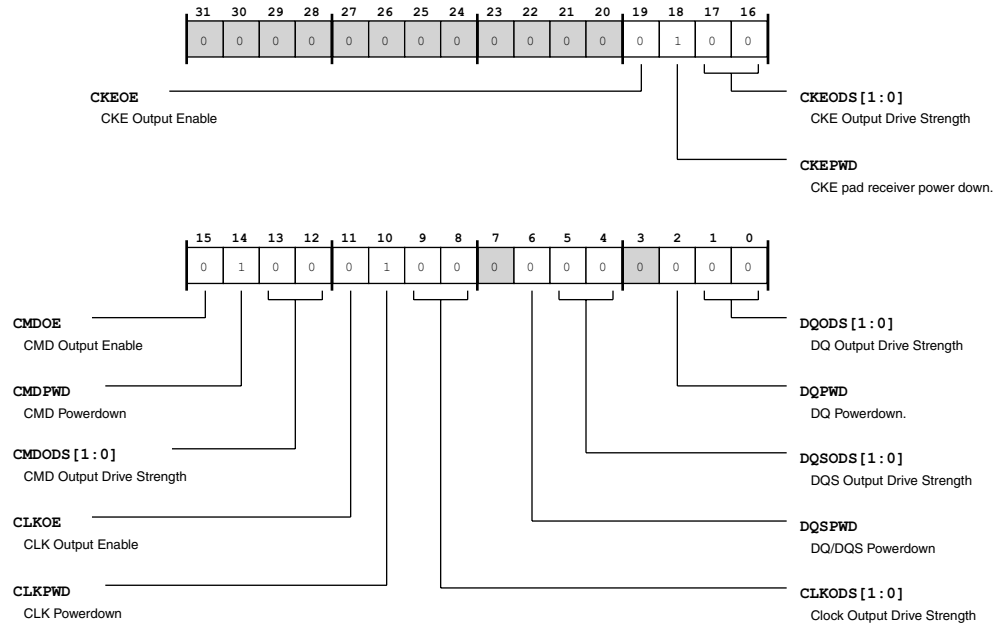


**Figure 11-19:** DMC\_PADCTL Register Diagram

**Table 11-20:** DMC\_PADCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------|
| 19      | CKEOE    | CKE Output Enable.<br>The DMC_PADCTL.CKEOE bit selects the CKE pad output enable value. |
|         |          | 0 Active OE value                                                                       |
|         |          | 1 Inactive OE value                                                                     |
| 18      | CKEPWD   | CKE pad receiver power down..                                                           |
|         |          | 0 Pad Receiver Powered Up                                                               |
|         |          | 1 Pad Receiver Powered Down                                                             |

**Table 11-20:** DMC\_PADCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 17:16   | CKEODS   | CKE Output Drive Strength.<br>The DMC_PADCTL.CKEODS bits select the CKE pads output drive strength.                                       |
|         |          | 0 SSTL18 full drive / LPDDR 10mA                                                                                                          |
|         |          | 1 SSTL 18 half drive / LPDDR 4mA                                                                                                          |
|         |          | 2 LPDDR 8mA - Reserved for DDR2 mode                                                                                                      |
|         |          | 3 LPDDR 2mA - Reserved for DDR2 mode                                                                                                      |
| 15      | CMDOE    | CMD Output Enable.<br>The DMC_PADCTL.CMDOE bit selects the CMD pads output enable value.                                                  |
|         |          | 0 Active OE Value                                                                                                                         |
|         |          | 1 Inactive OE Value                                                                                                                       |
| 14      | CMDPWD   | CMD Powerdown.<br>The DMC_PADCTL.CMDPWD bit selects whether the command, address, and control signal pads receiver is powered up or down. |
|         |          | 0 Pad Receiver Powered up                                                                                                                 |
|         |          | 1 Pad Receive Powered down                                                                                                                |
| 13:12   | CMDODS   | CMD Output Drive Strength.<br>The DMC_PADCTL.CMDODS bits select the command, address, and control signal drive strength.                  |
|         |          | 0 SSTL18 full drive / LPDDR 10mA                                                                                                          |
|         |          | 1 SSTL18 half drive / LPDDR 4mA                                                                                                           |
|         |          | 2 LPDDR 8 mA                                                                                                                              |
|         |          | 3 LPDDR 2 mA                                                                                                                              |

**Table 11-20:** DMC\_PADCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------|
| 11      | CLKOE    | CLK Output Enable.<br>The DMC_PADCTL.CLKOE bit selects the CLK pads output enable value.                           |
|         |          | 0                      Active OE Value                                                                             |
|         |          | 1                      Inactive OE Value                                                                           |
| 10      | CLKPWD   | CLK Powerdown.<br>The DMC_PADCTL.CLKPWD bit selects whether the CLK pads receiver is powered up or down.           |
|         |          | 0                      Pad Receiver Powered Up                                                                     |
|         |          | 1                      Pad Receiver Powered Down                                                                   |
| 9:8     | CLKODS   | Clock Output Drive Strength.<br>The DMC_PADCTL.CLKODS bits select clock pad output drive strength.                 |
|         |          | 0                      SSTL18 full drive / LPDDR 10mA                                                              |
|         |          | 1                      SSTL 18 half drive / LPDDR 4mA                                                              |
|         |          | 2                      LPDDR 8 mA                                                                                  |
|         |          | 3                      LPDDR 2 mA                                                                                  |
| 6       | DQSPWD   | DQ/DQS Powerdown.<br>The DMC_PADCTL.DQSPWD bit selects whether the DQ and DQS pads receiver is powered up or down. |
|         |          | 0                      Pad Receiver Powered Up                                                                     |
|         |          | 1                      Pad Receiver Powered Down                                                                   |

**Table 11-20:** DMC\_PADCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | DQSODS   | DQS Output Drive Strength.<br>The DMC_PADCTL.DQSODS bits select the DQS pads output drive strength. Note that DMC_PADCTL.DQSODS[3] is connected to S1 of PAD, and DMC_PADCTL.DQSODS[4] is connected to S0 of PAD.    |
|         |          | 0 SSTL18 full drive / LPDDR 10mA                                                                                                                                                                                     |
|         |          | 1 SSTL 18 half drive / LPDDR 4mA                                                                                                                                                                                     |
|         |          | 2 LPDDR 8mA - Reserved for DDR2 mode                                                                                                                                                                                 |
|         |          | 3 LPDDR 2mA                                                                                                                                                                                                          |
| 2       | DQPWD    | DQ Powerdown..<br>The DMC_PADCTL.DQPWD bit selects whether the DQ pads receiver is powered up or down.                                                                                                               |
|         |          | 0 Pad Receiver Powered Up                                                                                                                                                                                            |
|         |          | 1 Pad Receiver Powered Down                                                                                                                                                                                          |
| 1:0     | DQODS    | DQ Output Drive Strength.<br>The DMC_PADCTL.DQODS bits select the output drive strength for the DQ pads. Note that DMC_PADCTL.DQODS[0] is connected to A2 of PAD, and DMC_PADCTL.DQODS[1] is connected to A6 of PAD. |
|         |          | 0 ODT Disable                                                                                                                                                                                                        |
|         |          | 1 75 Ohm                                                                                                                                                                                                             |
|         |          | 2 150 Ohm                                                                                                                                                                                                            |



# 12 Cyclic Redundancy Check (CRC)

The CRC peripheral is used to perform the Cyclic Redundancy Check (CRC) of the block of data that is presented to the peripheral. The peripheral provides a means to periodically verify the integrity of the system memory, the contents of memory-mapped registers (MMRs), or communication message objects, and it is based on a CRC32 engine that computes the signature of 32-bit data presented to the peripheral.

The dedicated hardware compares the calculated signature of the operation to a pre-loaded expected signature and if the two fail to match, the peripheral generates an error.

Data may be provided by the source channel of the memory-to-memory DMA channels and optionally forwarded to memory via the destination DMA channel. Alternatively, the peripheral also supports data presented by core write transactions.

The CRC peripheral implements a reduced table-lookup algorithm to compute the signature of the data. A programmable 32-bit CRC polynomial is used to automatically generate the lookup table (LUT) contents.

Additional CRC peripheral modes allow for initializing large memory sections with a constant value, or for verifying that sections of memory are equal to a constant value.

## CRC Features

The CRC peripheral supports a number of key features, including memory scan modes for memory verification, memory transfer modes for on-the-fly CRC calculations while transferring data from one memory to another, a programmable 32-bit CRC polynomial with automatic LUT generation, and data mirroring options.

The CRC module includes the following features.

- CRC checksum computation and comparison modes
- 32-bit programmable CRC polynomial with bit reverse option
- Automatic look up table (LUT) generation
- Data mirroring options for endian and reflected polynomial cases
- Automatic clear and preset of results
- Fault and error interrupt reporting
- DMA and MMR based operation

Since the CRC module is closely tied to memory-to-memory DMA (MDMA) channel pairs, the use cases include the following features.

- Memory scan with CRC compute or compare
- Memory transfer with CRC compute or compare
- Memory fill with 32-bit data patterns
- Memory verify
- MMR write access to FIFO of destination DMA
- MMR read access to FIFO of source DMA
- Profiting from advanced DMA features, like descriptor mode and bandwidth control/monitor

## CRC Functional Description

The CRC peripheral supports a number of modes of operation that allows for the initialization and verification of regions of memory. The peripheral supports efficient memory fill and verification operations on regions of memory with or against a constant value. These modes of operation do not require the CRC engine to calculate a signature. Other modes of operation allow for the CRC signature to be calculated and verified for a memory region and also allow for on the fly CRC calculation when performing memory-to-memory DMA transfers from one memory region to another.

In order to minimize the need for core accesses, the peripheral is interfaced to memory-to-memory DMA (MDMA) channels, allowing for flexible configuration in which data may be written to or read from the peripheral via DMA transactions, core transactions, or a combination of both. Two DMA channels are supported, to provide both a data input and data output. CRC0 is connected to the MDMA0 channel pair and CRC1 is connected to the MDMA1 channel pair.

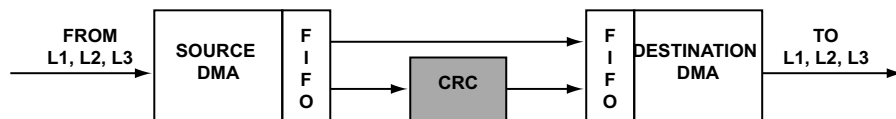


**Figure 12-1:** Memory Flow

The following sections describe in further detail the functional operation of the CRC peripheral:

- [CRC Register List](#)
- [CRC Definitions](#)
- [CRC Block Diagram](#)
- [CRC Architectural Concepts](#)

The CRC peripheral supports a number of modes of operation that allow for the initialization and verification of regions of memory. The peripheral supports efficient memory fill and verification of memory

regions with a constant value in which the modes of operation do not require the need to enable the CRC engine to calculate a signature. Other modes of operation allow for the CRC signature to be calculated and verified for a memory region or also to allow for the signature to be calculated and verified on the fly while doing a memory-to-memory DMA operation from one memory region to another.

## CRC Register List

The cyclic redundancy check (CRC) unit includes the data comparison, polynomial operation, and look up table generation features needed for CRC operation. The CRC provides CRC protection as specified by the ASIL (Automobile Safety Integrity Level) requirements for the ADAS (Advanced Driver Assistance System) segment. This unit meets the requirements that the system software should be able to periodically check the correctness of the code/data available in the memory. A set of registers govern CRC operations. For more information on CRC functionality, see the CRC register descriptions.

**Table 12-1:** CRC Module Registers

| Name           | Description                     |
|----------------|---------------------------------|
| CRC_CTL        | Control Register                |
| CRC_DCNT       | Data Word Count Register        |
| CRC_DCNTRLD    | Data Word Count Reload Register |
| CRC_COMP       | Data Compare Register           |
| CRC_FILLVAL    | Fill Value Register             |
| CRC_DFIFO      | Data FIFO Register              |
| CRC_INEN       | Interrupt Enable Register       |
| CRC_INEN_SET   | Interrupt Enable Set Register   |
| CRC_INEN_CLR   | Interrupt Enable Clear Register |
| CRC_POLY       | Polynomial Register             |
| CRC_STAT       | Status Register                 |
| CRC_DCNTCAP    | Data Count Capture Register     |
| CRC_RESULT_FIN | CRC Final Result Register       |
| CRC_RESULT_CUR | CRC Current Result Register     |

**Table 12-1:** CRC Module Registers (Continued)

| Name      | Description          |
|-----------|----------------------|
| CRC_REVID | Revision ID Register |

## ADSP-BF60x CRC Interrupt List

**Table 12-2:** ADSP-BF60x CRC Interrupt List

| Description               | ID | DMA Channel | Sensitivity |
|---------------------------|----|-------------|-------------|
| CRC0 Datacount expiration | 88 |             | LEVEL       |
| CRC0 Error                | 89 |             | LEVEL       |
| CRC1 Datacount expiration | 92 |             | LEVEL       |
| CRC1 Error                | 93 |             | LEVEL       |

## CRC Definitions

To make the best use of the CRC, it is useful to understand the following terms.

### CRC

Acronym for Cyclic Redundancy Check. An error detection code that is capable of detecting changes within a block of data.

### CRC Polynomial

The 32-bit polynomial used by the CRC engine to generate the Look-Up-Table required for the CRC implementation

### LUT

Acronym for the Look-Up-Table. The Look-Up-Table is automatically generated from the supplied 32-bit CRC polynomial.

### DMA

Acronym for Direct Memory Access. Used to describe a data transfer that takes place via a DMA channel allowing data to be distributed around a system without intervention from the core.

### MDMA

Acronym for Memory-To-Memory DMA transfer that often requires the use of two DMA channels to transfer data from one memory region to another memory region. One DMA channel is configured as a source channel and the second as a destination channel. Acronym for the Look-Up-Table. The Look-Up-Table is automatically generated from the supplied 32-bit CRC polynomial.

## CRC Block Diagram

The figure shows the functional block diagram of the CRC peripheral.

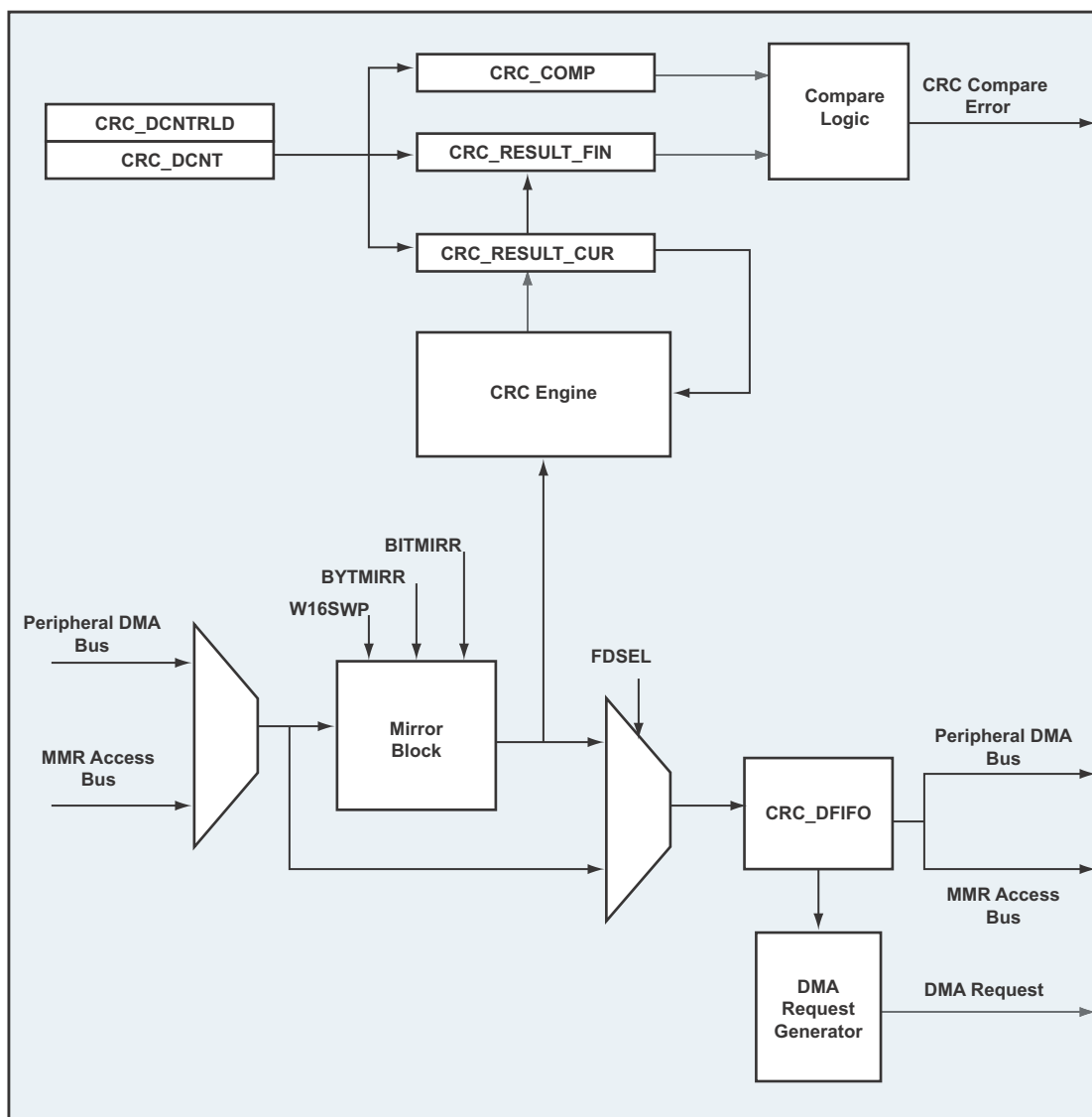


**Figure 12-2:** CRC Block Diagram

## Peripheral DMA Bus

The CRC peripheral provides both an incoming and outgoing data path to the Peripheral DMA Bus. The MDMA source channel is interfaced to the incoming data path providing data to the CRC peripheral. For memory transfer and data fill modes, the MDMA destination channel is used to either output the data from the CRC FIFO or the data to be used for the fill operation.

## MMR Access Bus

The MMR access bus is used by the core to access all the memory-mapped registers of the peripheral for configuration, status and debug purposes. The core may also use the MMR access bus to feed data to the

CRC peripheral or read data from the FIFO of the CRC peripheral as an alternative to the operation being performed by the DMA channels.

Data received by MMR writes can transfer to destination DMA. Similarly, data received by source DMA can be output MMR interface. Optionally, intermediate results can be made available to MMR interface.

## Compare Logic

The compare logic takes the final CRC signature and compares this to the expected CRC signature, generating a CRC compare error if the signatures do not match. A compare error can flag a system fault.

## Mirror Block

The mirror block individually controls bit reversing of the polynomial, the computation results and the expected result. Endian and reflection of processed data can be controlled by bit mirroring, byte mirroring, word swapping and any combination of thereof.

## CRC Data FIFO

The CRC Data FIFO is a 32-bit-wide 4-entry FIFO. The FIFO is accessible to both the Peripheral DMA bus and the MMR Access bus. The FIFO status is accessible from the `CRC_STAT` register.

## CRC Engine

The CRC Engine is a 32-bit CRC engine that implements the Reduced Table Lookup scheme. The CRC engine provides support for a user-programmable 32-bit polynomial that is used to load the lookup table parameters required for the CRC calculation. The CRC engine is a 2-cycle implementation operating on 16 bits of data per cycle.

## DMA Request Generator

The DMA Request Generator is responsible for granting incoming DMA requests from the source DMA channel and issuing outgoing DMA requests to the destination DMA channel.

## CRC Architectural Concepts

The CRC peripheral includes a 32-bit CRC engine that implements the reduced table lookup scheme operating on 16 bits of data per cycle, resulting in a 2-cycle implementation for each 32 bits of data written to the peripheral. The upper 16 bits of the data are processed in the first cycle, followed by the lower 16 bits.

A 32-bit polynomial is required before calculation of the CRC signature can occur. The polynomial is used to generate the contents of an internal lookup table that is required by the reduced table lookup implementation. The lookup table that is automatically generated when the polynomial is written must be initialized prior to any operation that requires the use of the CRC engine.

The data presented to the CRC engine may be manipulated by the mirror block logic before being used in the calculation of the CRC signature. The data mirror operation is configurable to allow for bit reversing, byte reversing and 16-bit word swapping operations to be applied to the incoming data. For memory transfer compute and compare operations, the user may configure the peripheral to output the data in the same form in which it was received, or output the mirrored data in the same manner that it is presented to the CRC engine.

While the CRC peripheral is in operation, the status of the FIFO is continually updated and reflected in the `CRC_STAT` register. The FIFO status is required for core-based accesses to the CRC peripheral. The status indicates when the CRC peripheral is capable of receiving data, when data is available to be read from the FIFO and when the result of the `CRC_RESULT_CUR` register has been updated, indicating that the current CRC calculation has completed and the result is available.

## Lookup Table

The lookup table consists of a set of 16 32-bit registers that are automatically populated by hardware when a write access takes place to the `CRC_POLY` register. 16 clock cycles are required to generate all 16 look up table entries. The status of the lookup table generation process is reflected in `CRC_STAT.LUTDONE`, allowing for software to poll on the completion of the event or for generation of an interrupt.

**NOTE:** The lookup table must be populated before any operation requiring the use of the CRC peripheral can take place, even if the operation does not require the use of the CRC engine. The peripheral will not issue any data requests until the table generation process has completed. In addition, the `CRC_STAT.IBR` field that indicates the input buffer status as required for core-based transfers is only valid upon completion of the lookup table generation process.

## Data Mirroring

The data mirror block may be configured to manipulate the incoming data before the data is passed on to the CRC engine and, optionally, to the FIFO. This allows the peripheral to handle various forms of endianness and to function with reflected polynomials.

There are three configuration bits that control the data mirroring process: `CRC_CTL.BITMIRR`, `CRC_CTL.BYTMIRR` and `CRC_CTL.W16SWP`. The following table details how these options affect the incoming data and the output that is generated by the mirror block.

**Table 12-3:** Data Mirroring Options

| W16SWP | BYTMIRR | BITMIRR | Output Data                                           |
|--------|---------|---------|-------------------------------------------------------|
| 0      | 0       | 0       | Dout[31:0] = Din[31:0]                                |
| 0      | 0       | 1       | Dout[31:0] = Din[24:31],Din[16:23],Din[8:15],Din[0:7] |
| 0      | 1       | 0       | Dout[31:0] = Din[7:0],Din[15:8],Din[23:16],Din[31:24] |



**Table 12-3:** Data Mirroring Options (Continued)

| W16SWP | BYTMIRR | BITMIRR | Output Data                                            |
|--------|---------|---------|--------------------------------------------------------|
| 0      | 1       | 1       | Dout[31:0] = Din[0:7],Din[8:15],Din[16:23],Din[24:31]  |
| 1      | 0       | 0       | Dout[31:0] = Din[15:0], D[31:16]                       |
| 1      | 0       | 1       | Dout[31:0] = Din[8:15],Din[0:7], Din[24:31],Din[16:23] |
| 1      | 1       | 0       | Dout[31:0] = Din[23:16],Din[31:24], Din[7:0],Din[15:8] |
| 1      | 1       | 1       | Dout[31:0] = Din[16:23],Din[24:31], Din[0:7],Din[8:15] |

When the CRC is configured to operate in the memory transfer compute and compare mode, the bit-reversed output data may be written to the FIFO. This feature is controlled via the `CRC_CTL.FDSEL` field.

In addition to providing bit swapping and mirror options to the incoming data, the CRC peripheral also supports bit mirroring on the following registers:

- `CRC_RESULT_CUR` and `CRC_RESULT_FIN`, controlled via the `CRC_CTL.RSLTMIRR` field. When mirroring is enabled, the values to be written to these registers are fully bit-reversed before being written.
- `CRC_POLY`, controlled via the `CRC_CTL.POLYMIRR` field. When mirroring is enabled, the 32-bit polynomial is fully bit-reversed before being written to the register.
- `CRC_COMP`, controlled via the `CRC_CTL.COMPMIRR` field. When mirroring is enabled, the contents to be loaded to this register are fully bit-reversed before being written.

## FIFO Status and Data Requests

The CRC peripheral provides input and output buffer status indication via `CRC_STAT.IBR` and `CRC_START.OBR`, respectively. For core-based operations, software is required to monitor these status fields prior to writing to or reading from the CRC FIFO. No write to the CRC FIFO should occur while `CRC_STAT.IBR` indicates that the buffer is not ready to accept data. Similarly, the CRC FIFO should not be read until `CRC_START.OBR` indicates that data is available.

The memory scan modes of operation only require the monitoring of the input buffer status, whereas the memory transfer compute and compare mode is required to use both input and output buffer status. If at any point the current result of the CRC computation is required, then software must verify that the current operation has completed and that the intermediate result is ready, as indicated by `CRC_STAT.IRR`.

**NOTE:** The memory transfer fill mode of operation requires the use of a DMA channel. Core reads from the CRC FIFO for this mode of operations are not supported.

Memory transfer compute and compare mode makes use of burst transactions in order to make the most efficient use of the available resources. In this mode, when the FIFO is initially empty and the peripheral is enabled, the `CRC_STAT.IBR` bit indicates that the CRC is ready to accept data, and the peripheral gener-

ates data requests to the source DMA channel (if DMA is used). As long as the number of words remaining in the `CRC_DATCNT` register is greater than the FIFO depth, the peripheral issues data requests or accepts incoming data in bursts until the CRC FIFO becomes full.

Once full, `CRC_STAT.IBR` and `CRC_STAT.OBR` bits are updated accordingly, and then outgoing data requests are issued. Only when the FIFO is empty will the peripheral then can accept further incoming data, and `CRC_STAT.IBR` and `CRC_STAT.OBR` bits are updated once again.

Once `CRC_DATCNT` is decremented such that the number of words remaining to be processed is less than the number of words required to fill the FIFO, the burst mode of operation is disabled and incoming data is accepted as long as the FIFO is not full and outgoing data is available as long the FIFO is not empty. Therefore, there are no restrictions requiring the word count to be a multiple of the FIFO depth.

All other CRC modes of operation indicate that incoming data may be accepted as long as the FIFO is not full, and that outgoing data is available as long the FIFO is not empty.

The way in which data requests and the status bits are generated is additionally influenced by the `CRC_CTL.OBRSTALL` and `CRC_CTL.IRRSTALL` bit configurations described below.

- The `CRC_CTL.OBRSTALL` bit may be configured such that the CRC peripheral stalls as soon as there is output data available in the FIFO. This mode of operation should only be used in memory transfer compute and compare mode. This results in the processing of a single 32-bit word at a time. The peripheral does not request or accept incoming data until the current value being processed is read from the peripheral.
- The `CRC_CTL.IIRSTALL` bit may be configured so that the CRC peripheral stalls all further incoming data requests until the `CRC_RESULT_CUR` register is read after being updated. This mode of operation is only used for modes that result in CRC signature generation. It is not applicable to memory transfer data fill or memory scan data verify modes of operation.

## CRC Operating Modes

The following sections describe the various operating modes of the CRC interface.

### Data Transfer Modes

The CRC peripheral supports two main categories of operation involving data transfers:

- Memory Scan mode
- Memory Transfer mode

Memory scan modes are read-only operations that allow the contents of memory to be read into the peripheral and verified for correctness. There are two forms of memory scan mode:

- CRC Compute and Compare performs a CRC calculation on data presented to the peripheral and compares the CRC result with a pre-determined and pre-loaded result. An error is generated if the results differ.
- Data Verify compares each 32-bit data word presented to the CRC peripheral to a pre-loaded 32-bit value and generates an error if the data is found to be different.

Both of these modes of operation require, at the very most, a single DMA channel to read the data from memory into the peripheral. No data is forwarded to data output or destination DMA. Core-driven transfers may also be used for either of these modes of operation.

The memory transfer modes, involve memory write or memory read and write operations allowing for memory to be initialized or transferred from one region of memory to another. There are two forms of memory transfer mode:

- CRC Compute and Compare performs a full data transfer from one memory region to another memory region. A CRC signature is generated on the data presented to the peripheral and compared with a pre-determined and pre-loaded result. An error is generated if the results differ.
- Data Fill initializes a region of memory with a pre-loaded 32-bit constant value.

The CRC compute and compare mode of operation requires both incoming and outgoing data channels either in the form of DMA channels, core driven write/read operations to/from the FIFO or a combination of both. The data fill mode of operation requires only a memory write DMA destination channel, this mode does not support core driven operations.

## Memory Scan Compute and Compare

In this mode of operation the CRC Engine of the peripheral is enabled. The mode is configured through the `CRC_CTL.OPMODE` field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured via the `CRC_DCNT` register. The accumulated result of the CRC operation is contained in the `CRC_RESULT_CUR` register. Upon each 32-bit word being processed by the CRC engine the `CRC_DCNT` register is decremented and `CRC_RESULT_CUR` is updated.

Once `CRC_DCNT` decrements to zero, the contents of the `CRC_RESULT_CUR` register are copied to `CRC_RESULT_FIN` and `CRC_STAT.DCNTEXP` is updated accordingly. The `CRC_COMP` register is used to store the expected result of the CRC operation. Upon completion of the CRC calculation, `CRC_COMP` is compared with `CRC_RESULT_FIN` and `CRC_STAT.CMPERR` is updated to reflect the status of the compare operation. `CRC_STAT.CMPERR` is required to be cleared before the next CRC operation is performed.

The CRC peripheral also contains `CRC_DCNTRLD` register. This register is used to reload `CRC_DCNT` upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation may be configured via `CRC_CTL.AUTOCCLRZ` and `CRC_CTL.AUTOCCLR`. This provides a means to reset `CRC_RESULT_CUR` to `0x00000000`, `0xFFFFFFFF` or to leave the current register contents untouched for the next operation.

The peripheral may be configured to allow for the compare error and data expiration events to generate an interrupt.

## Memory Scan Data Verify

In this mode of operation the CRC engine of the peripheral is not required. The mode is enabled through the `CRC_CTL.OPMODE` field. Each 32-bit word of the data stream is compared with a constant value that is stored in the `CRC_COMP` register. The `CRC_DCNT` register contains the number of words that are to be compared. The `CRC_DCNT` register is decremented upon receiving a new 32-bit word from the data stream. If at any point the compare operation should fail the `CRC_STAT.CMPERR` bit updated accordingly and the contents of `CRC_DCNT` are captured in the `CRC_DCNTCAP` register. This may be used in order to identify the location in the data stream where the error occurred. The `CRC_STAT.CMPERR` field should be cleared in order to re-enable capturing of further errors.

Once `CRC_DCNT` decrements to zero, `CRC_STAT.DCNTEXP` is updated accordingly to signal the end of the operation. The peripheral may be configured to allow for the compare error and data expiration events to generate an interrupt.

## Memory Transfer Compute and Compare

In this mode of operation the CRC Engine of the peripheral is enabled. The mode is configured through the `CRC_CTL.OPMODE` field and the CRC engine performs a 32-bit CRC operation on the incoming data stream.

The length of the data stream is configured via the `CRC_DCNT` register. The accumulated result of the CRC operation is contained in the `CRC_RESULT_CUR` register. Upon each 32-bit word being processed by the CRC engine the `CRC_DCNT` register is decremented and `CRC_RESULT_CUR` is updated.

Once `CRC_DCNT` decrements to zero, the contents of the `CRC_RESULT_CUR` register are copied to `CRC_RESULT_FIN` and `CRC_STAT.DCNTEXP` is updated accordingly. The `CRC_COMP` register is used to store the expected result of the CRC operation. Upon completion of the CRC calculation, `CRC_COMP` is compared with `CRC_RESULT_FIN` and `CRC_STAT.CMPERR` is updated to reflect the status of the compare operation. `CRC_STAT.CMPERR` is required to be cleared before the next CRC operation is performed.

The CRC peripheral also contains `CRC_DCNTRLD` register. This register is used to reload `CRC_DCNT` upon completion of the CRC operation in preparation for the next transfer.

The initial seed of the CRC computation may be configured via `CRC_CTL.AUTOCCLRZ` and `CRC_CTL.AUTOCCLR`. This provides a means to reset `CRC_RESULT_CUR` to `0x00000000`, `0xFFFFFFFF` or to leave the current register contents untouched for the next operation.

The peripheral may be configured to allow for the compare error and data expiration events to generate an interrupt.

## Memory Transfer Data Fill Mode

In this mode of operation the CRC engine of the peripheral is not required. The mode is enabled through the `CRC_CTL.OPMODE` field. The `CRC_FILLVAL` register is written with a 32-bit value. This value is used to initialize a block memory via the Memory-to-Memory DMA Destination channel. When the CRC peripheral and the DMA destination channel are enabled, the contents of the `CRC_FILLVAL` register is written to the DMA channel to initialize the memory region. The `CRC_DCNT` register contains the number of words that are to be written.

Once `CRC_DCNT` decrements to zero, `CRC_STAT.DCNTEXP` is updated accordingly to signal the end of the operation. The peripheral may be configured to allow for the this data expiration event to generate an interrupt.

## CRC Event Control

The CRC peripheral has options to enable certain CRC status operations to generate an interrupt event to the System Event Controller. There, a CRC error can be qualified as a system fault.

## Interrupt Signals

The CRC peripheral is capable of generating two interrupts that may optionally be enabled within the System Event Controller. One is a CRC status interrupt and the other a CRC error interrupt.

The `CRC_STAT.CMPERR` status bit may be configured as an interrupt as is signalled via the CRC error interrupt signal. The `CRC_STAT.CMPERR` status field is set whenever a compare operation performed by the CRC peripheral fails. This may be as the result of a Memory scan data verify operation failing that compares the contents of a memory range with a constant 32-bit value, or it may be as a result of the CRC signature calculated for a memory region not matching the expected pre-programmed result for a memory scan or memory transfer compute compare operation.

The `CRC_STAT.DCNTEXP` status bit is set when the `CRC_DCNT` register has decremented to zero indicating that the CRC peripheral has now processed all the data that was requested for the current CRC operation. This signal may also be used to generate an interrupt. The interrupt is signalled on the CRC Status interrupt signal.

Both these status bits may be configured to generate an interrupt via the `CRC_INEN` register. `CRC_INEN` register also has bit set, `CRC_INEN_SET`, and bit clear `CRC_INEN_CLR` equivalent registers that may be used for the enabling and disabling of these interrupt sources.

The `CRC_STAT` register has two write one to clear (W1C) fields for clearing the two interrupt sources.

**NOTE:** Disabling the CRC peripheral via `CRC_CTL.BLKEN` does not result in the interrupt sources being cleared. The interrupt sources must be cleared via a W1C operation to `CRC_STAT`.

## CRC Programming Model

It is important to note the following restrictions when using the CRC peripheral in conjunction with the DMA channels:

1. When enabling the CRC peripheral and the DMA channels, the CRC peripheral should be enabled prior to enabling the DMA channels.
2. When disabling the CRC peripheral and the DMA channels, the DMA channels should be disabled prior to disabling the CRC peripheral.

## CRC Mode Configuration

Describes a number of tasks showing the various operation modes of the CRC peripheral.

- *Look-Up Table Generation*
- *Core Driven Memory Scan Compute Compare Mode*
- *DMA Driven Memory Scan Compute Compare Mode*
- *Core Driven Memory Scan Data Verify Mode*
- *DMA Driven Memory Scan Data Verify Mode*
- *Core Driven Memory Transfer Compute Compare Mode*
- *DMA Driven Memory Transfer Compute Compare Mode*
- *DMA Driven Memory Transfer Data Fill Mode*

## Look-Up Table Generation

Describes the steps required to initialize the CRC peripheral Look-Up Table.

1. Write the 32-bit CRC polynomial of choice to `CRC_POLY`

*ADDITIONAL INFORMATION:* This operation results in the CRC peripheral starting the Look-Up Table initialization process. `CRC_STAT.LUTDONE` is updated to reflect the operation is in progress.

2. Poll `CRC_STAT.LUTDONE` until the status bit indicates that the operation is completed.

*RESULT:* The CRC peripheral has completed initialization of all the Look-Up Table registers and is now ready for data operations. The `CRC_STAT.LUTDONE` field remains in the current state until `CRC_POLY` is written again, or the peripheral or processor are reset.

## Core Driven Memory Scan Compute Compare Mode

Performs CRC signature calculation and verification for a region of memory using core transactions. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured via `CRC_CTL` being disabled.

### *PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserved CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`.

#### 1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

#### 2. Initialize `CRC_DCNTRLD`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

#### 3. Initialize `CRC_RESULT_CUR`.

*ADDITIONAL INFORMATION:* This register may be initialized to provide an initial seed for the CRC operation that is about to take place.

#### 4. Initialize `CRC_COMP`.

*ADDITIONAL INFORMATION:* This register contains the pre-calculated final CRC signature result for the memory region that is used in the final compare operation.

#### 5. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

#### 6. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to Memory Scan Compute Compare Mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

- `CRC_CTL.OBRSTALL` and `CRC_CTL.IRRSTALL` options must be disabled for this task example.
- All mirroring and bit reversal options should also be configured.
- CRC auto clear options should also be configured.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

7. Write memory region data to the CRC peripheral.

a. While `CRC_STAT.IBR` indicates input buffer is ready, write `CRC_FIFO` with 32-bit data.

*ADDITIONAL INFORMATION:* This step is repeated until all required data has been written.

8. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

9. Poll `CRC_STAT.CMPERR` if the interrupt was disabled to check for a compare error.

*ADDITIONAL INFORMATION:* This step is only required if the compare error interrupt is not enabled.

10. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC Compute Compare operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation.

*RESULT:*

The integrity check of the memory via the expected CRC signature has completed and the final result is indicated via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits are cleared before performing a further CRC operation.

## DMA Driven Memory Scan Compute Compare Mode

Performs CRC signature calculation and verification for a region of memory using DMA transactions. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured via `CRC_CTL` being disabled.

*PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserviced CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.



2. Initialize CRC\_DCNTRLD.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the CRC\_DCNT register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize CRC\_RESULT\_CUR.

*ADDITIONAL INFORMATION:* This register may be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize CRC\_COMP.

*ADDITIONAL INFORMATION:* This register contains the pre-calculated final CRC signature result for the memory region that is used in the final compare operation.

5. Initialize CRC\_INEN.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

6. Initialize CRC\_CTL with CRC\_CTL.OPMODE set to memory scan compute compare mode and CRC\_CTL.BLKEN configured to enable the CRC peripheral.

- CRC\_CTL.OBRSTALL and CRC\_CTL.IRRSTALL options must be disabled for this task example.
- All mirroring and bit reversal options should also be configured.
- CRC auto clear options should also be configured.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

7. Configure and enable the Memory-to-Memory Source DMA channel for memory read STOP mode.

*ADDITIONAL INFORMATION:* This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

8. Poll CRC\_STAT.DCNTEXP if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

9. Poll CRC\_STAT.CMPERR if the interrupt was disabled to check for a compare error.

*ADDITIONAL INFORMATION:* This step is only required if the compare error interrupt is not enabled.

10. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC Compute Compare operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation..

*STEP RESULT:*

*RESULT:*

The integrity check of the memory via the expected CRC signature has completed and the final result indicated is via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits are cleared before performing a further CRC operation. Any `W1C` status bits of the Memory-to-Memory Source DMA channel should also be cleared before the next CRC operation.

## Core Driven Memory Scan Data Verify Mode

Reads a region of memory using core transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

*PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserved CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`. The interrupt service routine for the compare error interrupt should read and store the contents of `CRC_DCNTCAP` to a buffer before clearing the compare error interrupt.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

2. Initialize `CRC_DCNTRLD`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize `CRC_COMP`.

*ADDITIONAL INFORMATION:* This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32-bit of data presented to the peripheral will be compared with this value.

4. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

5. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to Memory Scan Data Verify Mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

6. Write memory region data to the CRC peripheral.

- a. Poll `CRC_STAT.IBR` until input buffer is ready.
- b. Write `CRC_FIFO` with 32-bit data.

*ADDITIONAL INFORMATION:*

These two steps are repeated until the entire memory region has been written to the CRC peripheral.

7. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

8. Check if the buffer used to capture the `CRC_DCNTCAP` register upon a compare error has any new entries.

*ADDITIONAL INFORMATION:* The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC Memory Scan Verify operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation.

*STEP RESULT:*

*RESULT:*

The result of the integrity check of the memory with the 32-bit constant is indicated via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled. Each comparison error is traceable due to the logging of `CRC_DCNTCAP` from within the compare error interrupt handler.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits are cleared before performing a further CRC operation.

## DMA Driven Memory Scan Data Verify Mode

Reads a region of memory using DMA transactions and performs a compare operation on each 32-bit word against a single pre-loaded 32-bit constant. The compare error interrupt is enabled to capture and log the location of any compare errors.

*PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserved CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`. The interrupt service routine for the compare error interrupt should read and store the contents of `CRC_DCNTCAP` to a buffer before clearing the compare error interrupt.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

2. Initialize `CRC_DCNTRL`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize `CRC_COMP`.

*ADDITIONAL INFORMATION:* This register contains the 32-bit constant that the memory region is expected to be filled with. Each 32-bit of data presented to the peripheral will be compared with this value.

4. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

5. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to memory scan data verify mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

6. Configure and enable the memory-to-memory source DMA channel for memory read STOP mode.

*ADDITIONAL INFORMATION:* This step starts the data transfer from the memory region and writes the data to the CRC peripheral.

7. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

8. Check if the buffer used to capture the `CRC_DCNTCAP` register upon a compare error has any new entries.

*ADDITIONAL INFORMATION:* The values captures in the buffer provide a means to locate where in the memory region the failures occurred.

9. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC Memory Scan Verify operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation.

*STEP RESULT:*

*RESULT:*

The result of the integrity check of the memory with the 32-bit constant is indicated via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled. Each comparison error is traceable due to the logging of `CRC_DCNTCAP` from within the compare error interrupt handler.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits and DMA status bits are cleared before performing a further CRC operation.

## Core Driven Memory Transfer Compute Compare Mode

Performs CRC signature calculation and verification for a region of memory using core transactions while copying the contents to another memory region. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured via `CRC_CTL` being disabled.

### *PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserved CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

2. Initialize `CRC_DCNTRL`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize `CRC_RESULT_CUR`.

*ADDITIONAL INFORMATION:* This register may be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize `CRC_COMP`.

*ADDITIONAL INFORMATION:* This register contains the pre-calculated final CRC signature result for the memory region that is used in the final compare operation.

5. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

6. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to Memory Transfer Compute Compare Mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

- `CRC_CTL.OBRSTALL` and `CRC_CTL.IRRSTALL` options must be disabled for this task example.
- All mirroring and bit reversal options should also be configured.
- CRC auto clear options should also be configured.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

7. Write memory region data to the CRC peripheral and read it back to the new destination.
  - a. While `CRC_STAT.IBR` indicates input buffer is ready, write `CRC_FIFO` with 32-bit data.
  - b. While `CRC_STAT.OBR` indicates output buffer is ready, read `CRC_FIFO` and store data to new destination.

*ADDITIONAL INFORMATION:* These two steps are repeated until all required data has been processed through the CRC peripheral and copied to the new destination.

8. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if the counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

9. Poll `CRC_STAT.CMPERR` if the interrupt was disabled to check for a compare error.

*ADDITIONAL INFORMATION:* This step is only required if the compare error interrupt is not enabled.

10. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC compute and compare operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

#### *RESULT:*

The memory region has been copied to a new location and an integrity check of the memory via the expected CRC signature has also completed and the final result is indicated via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled.

#### *AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits are cleared before performing a further CRC operation.

## **DMA Driven Memory Transfer Compute Compare Mode**

Performs CRC signature calculation and verification for a region of memory using DMA transactions. The memory region is also copied to another memory region via the use of Memory-to-Memory DMA trans-

fers. The CRC peripheral is configured such that it operates in the burst mode of operation due to the stalling options configured via `CRC_CTL` being disabled.

*PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserviced CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

2. Initialize `CRC_DCNTRL`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize `CRC_RESULT_CUR`.

*ADDITIONAL INFORMATION:* This register may be initialized to provide an initial seed for the CRC operation that is about to take place.

4. Initialize `CRC_COMP`.

*ADDITIONAL INFORMATION:* This register contains the pre-calculated final CRC signature result for the memory region that is used in the final compare operation.

5. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

6. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to Memory Transfer Compute Compare Mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

- `CRC_CTL.OBRSTALL` and `CRC_CTL.IRRSTALL` options must be disabled for this task example.
- All mirroring and bit reversal options should also be configured.
- CRC auto clear options should also be configured.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.



7. Configure and enable the Memory-to-Memory Source DMA channel for memory read STOP mode and destination DMA channel for memory write STOP mode.

*ADDITIONAL INFORMATION:* This step starts the data transfer from one memory region to another via the Memory-to-Memory DMA channels and the CRC peripheral.

8. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

9. Poll `CRC_STAT.CMPERR` if the interrupt was disabled to check for a compare error.

*ADDITIONAL INFORMATION:* This step is only required if the compare error interrupt is not enabled.

10. Write `CRC_STAT` to clear both `CRC_STAT.DCNTEXP` and `CRC_STAT.CMPERR`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of these status bits should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC compute and compare operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation. The memory region has also been copied to its new destination.

*RESULT:*

The integrity check of the memory via the expected CRC signature has completed and the final result is indicated via `CRC_STAT.CMPERR` and the corresponding interrupt if it were enabled. The memory region has also been copied to its final destination.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits are cleared before performing a further CRC operation. Any `W1C` status bits of the Memory-to-Memory Source and Destination DMA channels should also be cleared before the next CRC operation.

## DMA Driven Memory Transfer Data Fill Mode

Initializes a region of memory to a constant 32-bit value using DMA transactions.

### *PREREQUISITE:*

The task assumes that the polynomial has been loaded and the look-up table is fully initialized, there are no previously unserved CRC interrupts pending, and the CRC block is currently disabled as per `CRC_CTL.BLKEN`.

1. Initialize `CRC_DCNT`.

*ADDITIONAL INFORMATION:* The value loaded must represent the number of 32-bit words in the memory region for which the CRC signature is to be calculated and verified.

2. Initialize `CRC_DCNTRL`.

*ADDITIONAL INFORMATION:* This is the value that is used to reload the `CRC_DCNT` register upon completion of current CRC operation. If no further operation is required then this register may be initialized to zero.

3. Initialize `CRC_FILLVAL`.

*ADDITIONAL INFORMATION:* This register contains the 32-bit constant that is used to fill the memory region.

4. Initialize `CRC_INEN`.

*ADDITIONAL INFORMATION:* This register is used to enable the generation of the CRC interrupts for notification of compare errors and block completion. Configure these interrupts as required. If enabled ensure the corresponding interrupt handlers are also configured.

5. Initialize `CRC_CTL` with `CRC_CTL.OPMODE` set to Memory Transfer Fill Mode and `CRC_CTL.BLKEN` configured to enable the CRC peripheral.

*STEP RESULT:* The CRC peripheral is now enabled and ready for data to be written by the core or DMA channel.

6. Configure and enable the Memory-to-Memory Destination DMA channel for memory write STOP mode.

*ADDITIONAL INFORMATION:* This step starts the data transfer taking the constant 32-bit value from the CRC peripheral and writing the data to the DMA channel.

7. Poll `CRC_STAT.DCNTEXP` if the interrupt was disabled.

*ADDITIONAL INFORMATION:* This step is required only if counter expired interrupt is disabled. Polling is required to ensure all the data has been processed.

8. Write `CRC_STAT` to clear `CRC_STAT.DCNTEXP`.

*ADDITIONAL INFORMATION:* If interrupts were enabled then the clearing of this status bit should be performed within the interrupt handlers for the respective interrupts.

*STEP RESULT:* The CRC Memory Transfer Fill operation is now complete and the CRC peripheral is ready to be configured for the next CRC operation.

*RESULT:*

The memory region is now filled with the constant data and the CRC peripheral is ready to be configured for a new operation.

*AFTER COMPLETING THIS TASK:*

Ensure any `W1C` CRC status bits and DMA status bits are cleared before performing a further CRC operation.

## CRC Peripheral and DMA Channel List

**Table 12-4:** CRC DMA Channels

| DMA Channel | Peripheral    | FIFO Depth (Bytes) | Bandwidth Limit/Monitor Support |
|-------------|---------------|--------------------|---------------------------------|
| DMA21       | CRC0 Receive  | 128                | Yes                             |
| DMA22       | CRC0 Transmit | 64                 | Yes                             |
| DMA23       | CRC1 Receive  | 64                 | Yes                             |
| DMA24       | CRC1 Transmit | 64                 | Yes                             |

**Table 12-5:** CRC DMA Channels (Continued)

| DMA Channel | Memory Bus Width | Peripheral Bus Width | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------------|----------------------|-----------------------|------------------------|
| DMA21       | 32-bit           | 32-bit               | 8                     | 7                      |
| DMA22       | 32-bit           | 32-bit               | 8                     | 4                      |
| DMA23       | 32-bit           | 32-bit               | 8                     | 4                      |
| DMA24       | 32-bit           | 32-bit               | 8                     | 4                      |

## CRC Register Descriptions

Cyclic Redundancy Check Unit (CRC) contains the following registers.

**Table 12-6:** CRC Module Registers

| Name           | Description                     |
|----------------|---------------------------------|
| CRC_CTL        | Control Register                |
| CRC_DCNT       | Data Word Count Register        |
| CRC_DCNTRLD    | Data Word Count Reload Register |
| CRC_COMP       | Data Compare Register           |
| CRC_FILLVAL    | Fill Value Register             |
| CRC_DFIFO      | Data FIFO Register              |
| CRC_INEN       | Interrupt Enable Register       |
| CRC_INEN_SET   | Interrupt Enable Set Register   |
| CRC_INEN_CLR   | Interrupt Enable Clear Register |
| CRC_POLY       | Polynomial Register             |
| CRC_STAT       | Status Register                 |
| CRC_DCNTCAP    | Data Count Capture Register     |
| CRC_RESULT_FIN | CRC Final Result Register       |
| CRC_RESULT_CUR | CRC Current Result Register     |
| CRC_REVID      | Revision ID Register            |

### Control Register

The CRC\_CTL configures the operation modes and settings for the CRC.

CRC\_CTL: Control Register - R/W

Reset = 0x0000 0000

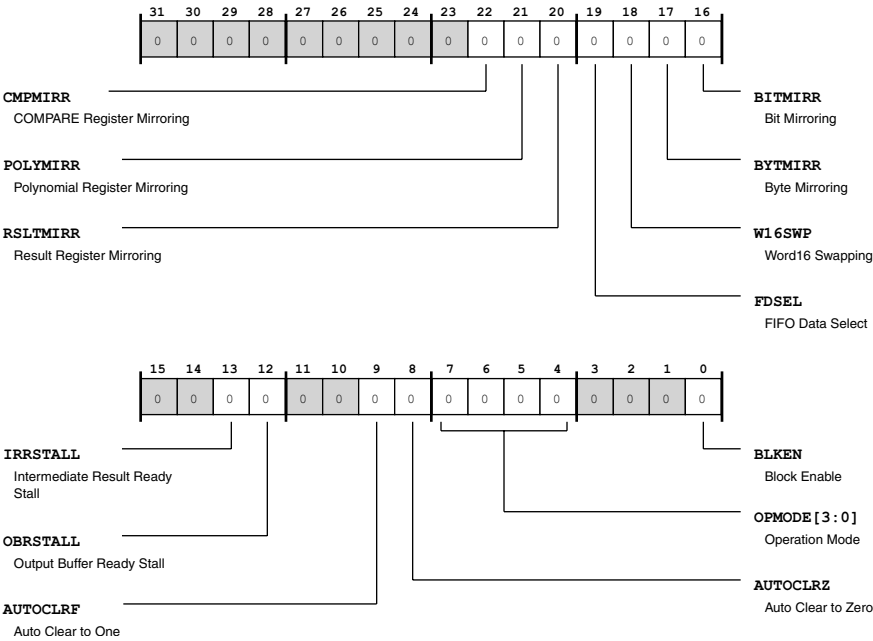


Figure 12-3: CRC\_CTL Register Diagram

Table 12-7: CRC\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | CMPMIRR  | COMPARE Register Mirroring.<br>The CRC_CTL.CMPMIRR enables data mirroring for the CRC_COMP compare register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for comparison with the CRC_RESULT_FIN register. |
|         |          | 0 Disable compare mirroring                                                                                                                                                                                                                                                    |
|         |          | 1 Enable compare mirroring                                                                                                                                                                                                                                                     |

**Table 12-7:** CRC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 21      | POLYMIRR | Polynomial Register Mirroring.<br>The <code>CRC_CTL.POLYMIRR</code> enables data mirroring for the <code>CRC_POLY</code> polynomial register. When enabled, the 32-bit value in this register is fully bit mirrored (reversed). The bit-reversed value is used for CRC computations.                                                                                           |
|         |          | 0      Disable polynomial mirroring                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Enable polynomial mirroring                                                                                                                                                                                                                                                                                                                                             |
| 20      | RSLTMIRR | Result Register Mirroring.<br>The <code>CRC_CTL.RSLTMIRR</code> enables data mirroring for the <code>CRC_RESULT_CUR</code> and <code>CRC_RESULT_FIN</code> result registers. When enabled, the 32-bit values in these registers are fully bit mirrored (reversed).                                                                                                             |
|         |          | 0      Disable result mirroring                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable result mirroring                                                                                                                                                                                                                                                                                                                                                 |
| 19      | FDSEL    | FIFO Data Select.<br>The <code>CRC_CTL.FDSEL</code> selects whether the CRC writes modified or unmodified data to the FIFO in memory transfer mode. If enabled, the data written is affected by the state of the data mirroring selections ( <code>CRC_CTL.BITMIRR</code> , <code>CRC_CTL.BYTMIRR</code> , and <code>CRC_CTL.W16SWP</code> ) before being written to the FIFO. |
|         |          | 0      Write unmodified data to FIFO                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1      Write modified data to FIFO                                                                                                                                                                                                                                                                                                                                             |
| 18      | W16SWP   | Word16 Swapping.<br>The <code>CRC_CTL.W16SWP</code> enables the CRC's data mirror block to swap the upper and lower 16-bit words within the 32-bit input data, before further processing.                                                                                                                                                                                      |
|         |          | 0      Disable word16 swapping                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1      Enable word16 swapping                                                                                                                                                                                                                                                                                                                                                  |

**Table 12-7:** CRC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | BYTMIRR  | Byte Mirroring.<br>The <code>CRC_CTL.BYTMIRR</code> enables the CRC's data mirror block to mirror the bytes within the 32-bit input data, before further processing.                                                                                                                                                                         |
|         |          | 0      Disable byte mirroring                                                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable byte mirroring                                                                                                                                                                                                                                                                                                                 |
| 16      | BITMIRR  | Bit Mirroring.<br>The <code>CRC_CTL.BITMIRR</code> enables the CRC's data mirror block to mirror the bits within each byte of the 32-bit input data, before further processing.                                                                                                                                                              |
|         |          | 0      Disable bit mirroring                                                                                                                                                                                                                                                                                                                 |
|         |          | 1      Enable bit mirroring                                                                                                                                                                                                                                                                                                                  |
| 13      | IRRSTALL | Intermediate Result Ready Stall.<br>The <code>CRC_CTL.IRRSTALL</code> enables stalling the state machine for input data when there is a valid intermediate result to be read in <code>CRC_RESULT_CUR</code> . This feature should be used only in CRC computation modes (for example, <code>CRC_CTL.OPMODE = 1</code> or <code>= 3</code> ). |
|         |          | 0      Do not stall                                                                                                                                                                                                                                                                                                                          |
|         |          | 1      Stall on IRR                                                                                                                                                                                                                                                                                                                          |
| 12      | OBRSTALL | Output Buffer Ready Stall.<br>The <code>CRC_CTL.OBRSTALL</code> enables stalling the state machine for input data when there is a valid data in the output buffer. This feature should be used only in memory-to-memory transfer modes (for example, <code>CRC_CTL.OPMODE = 1</code> ).                                                      |
|         |          | 0      Do not stall                                                                                                                                                                                                                                                                                                                          |
|         |          | 1      Stall on OBR                                                                                                                                                                                                                                                                                                                          |

**Table 12-7:** CRC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | AUTOCLRF | Auto Clear to One.<br>The <code>CRC_CTL.AUTOCLRF</code> enables auto clear to one when the CRC is in intermediate results ready stall mode ( <code>CRC_CTL.IRRSTALL=1</code> ) and the CRC data count expires ( <code>CRC_DCNT=0</code> ). Note that <code>CRC_CTL.AUTOCLRZ</code> must be disabled, or the <code>CRC_CTL.AUTOCLRF</code> has no effect.   |
|         |          | 0 No auto clear                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Auto clear                                                                                                                                                                                                                                                                                                                                               |
| 8       | AUTOCLRZ | Auto Clear to Zero.<br>The <code>CRC_CTL.AUTOCLRZ</code> enables auto clear to zero when the CRC is in intermediate results ready stall mode ( <code>CRC_CTL.IRRSTALL=1</code> ) and the CRC data count expires ( <code>CRC_DCNT=0</code> ). Note that <code>CRC_CTL.AUTOCLRF</code> must be disabled, or the <code>CRC_CTL.AUTOCLRZ</code> has no effect. |
|         |          | 0 No auto clear                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Auto clear                                                                                                                                                                                                                                                                                                                                               |
| 7:4     | OPMODE   | Operation Mode.<br>The <code>CRC_CTL.OPMODE</code> selects the memory transfer or scan mode.                                                                                                                                                                                                                                                               |
|         |          | 0 Reserved                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 CRC compute/compare memory transfer                                                                                                                                                                                                                                                                                                                      |
|         |          | 2 Data fill memory transfer                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 CRC compute/compare memory scan                                                                                                                                                                                                                                                                                                                          |
|         |          | 4 Data verify memory scan                                                                                                                                                                                                                                                                                                                                  |
| 0       | BLKEN    | Block Enable.<br>The <code>CRC_CTL.BLKEN</code> enables/disables CRC operation.                                                                                                                                                                                                                                                                            |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                   |



## Data Word Count Register

The `CRC_DCNT` holds the word count that is used for the CRC operation. On transfer of every 32-bit word, the CRC decrements by 1 the content of this register. When the count decrements to zero, this event triggers a CRC compare action, and `CRC_DCNT` is automatically loaded from the `CRC_DCNTRLD` for the next CRC operation. Note that the initial value programmed into `CRC_DCNT` may be different from what is programmed in the `CRC_DCNTRLD`.

**CRC\_DCNT: Data Word Count Register - R/W**

Reset = 0x0000 0000

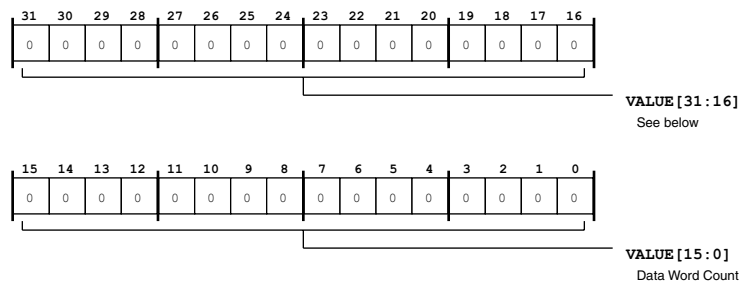


**Figure 12-4:** CRC\_DCNT Register Diagram

**Table 12-8:** CRC\_DCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Data Word Count.        |

## Data Word Count Reload Register

The `CRC_DCNTRLD` holds the value that the CRC automatically loads into `CRC_DCNT` when the `CRC_DCNT` decrements to 0. At startup, the value programmed in `CRC_DCNT` and `CRC_DCNTRLD` could be different. So, for the first iteration, the CRC operation happens for the count initially programmed in the `CRC_DCNT` register. While for subsequent CRC operations, the count is taken from the `CRC_DCNTRLD` register.

CRC\_DCNTRLD: Data Word Count Reload Register - R/W

Reset = 0x0000 0000

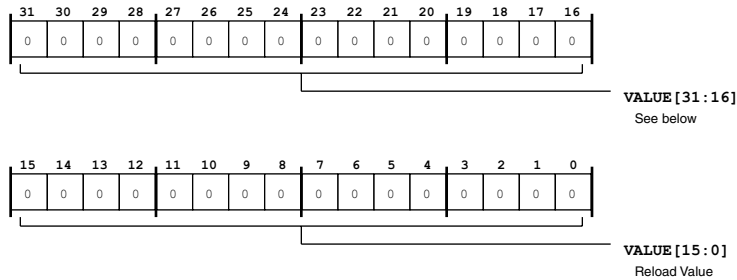


Figure 12-5: CRC\_DCNTRLD Register Diagram

Table 12-9: CRC\_DCNTRLD Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Reload Value.           |

Data Compare Register

The CRC\_COMP contains the value corresponding to the expected CRC result or signature for the current data stream. At the end of the operation, the content of this register is used to compare against the result produced by the CRC operation. In data verify mode, each incoming data value is compared with the content of this register.

CRC\_COMP: Data Compare Register - R/W

Reset = 0x0000 0000

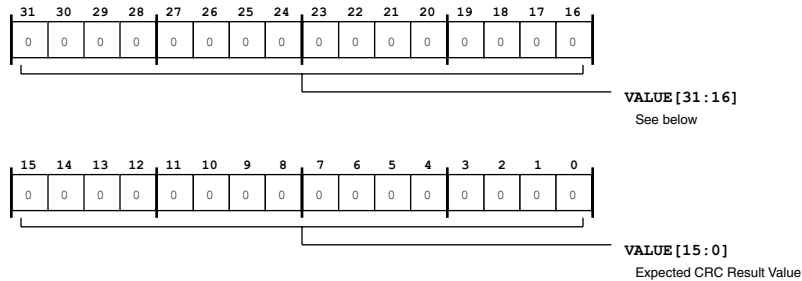


Figure 12-6: CRC\_COMP Register Diagram

**Table 12-10:** CRC\_COMP Register Fields

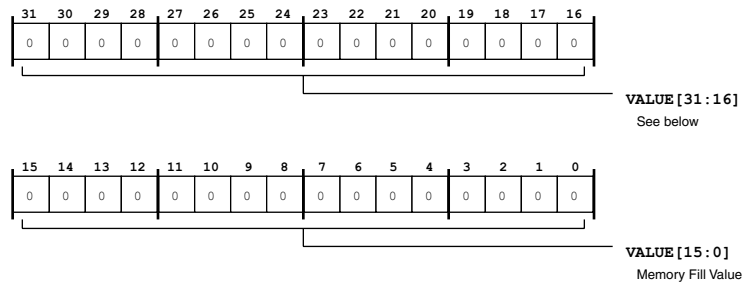
| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Expected CRC Result Value. |

## Fill Value Register

The `CRC_FILLVAL` holds the value that the CRC uses for the memory fill operation. In data fill mode, the value programmed in this register is used for the memory fill operation.

**CRC\_FILLVAL: Fill Value Register - R/W**

Reset = 0x0000 0000

**Figure 12-7:** CRC\_FILLVAL Register Diagram**Table 12-11:** CRC\_FILLVAL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Memory Fill Value.      |

## Data FIFO Register

In memory transfer mode (non-data fill mode), the data from the DMA or processor core buses is written into the `CRC_DFIFO` on each input data grant (DMA grant or core write). Data is read from this FIFO on each output data grant (DMA grant or core read). FIFO status information is available in the `CRC_STAT` register. Whenever, the FIFO has valid data, output data requests are generated.

Note that---in non-memory transfer mode and in data fill mode---the input data actually does not get written into this FIFO. So, this register should not be read in these modes.

CRC\_DFIFO: Data FIFO Register - R/W

Reset = 0x0000 0000

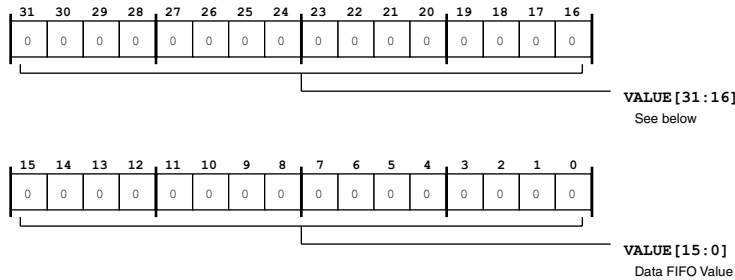


Figure 12-8: CRC\_DFIFO Register Diagram

Table 12-12: CRC\_DFIFO Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Data FIFO Value.        |

Interrupt Enable Register

The CRC\_INEN unmask (enables) or mask (disables) interrupt requests generated in the CRC from going to the processor core. Note that CRC interrupts are not disabled when the CRC is disabled (CRC\_CTL.BLKEN =0).

CRC\_INEN: Interrupt Enable Register - R/W

Reset = 0x0000 0000

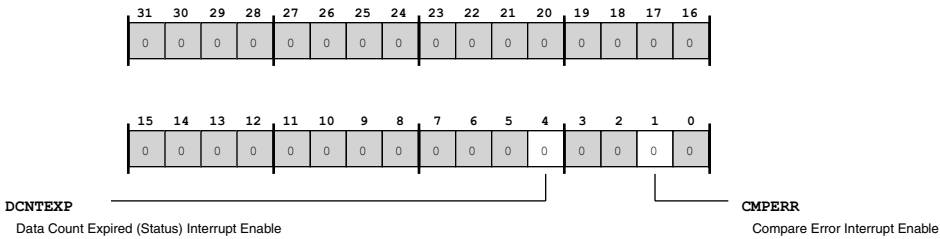


Figure 12-9: CRC\_INEN Register Diagram

Table 12-13: CRC\_INEN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DCNTEXP  | Data Count Expired (Status) Interrupt Enable.<br>The CRC_INEN.DCNTEXP enables (unmasks) the data count expired (CRC status) interrupt.                  |
|         |          | 0                      Disable (mask) interrupt                                                                                                         |
|         |          | 1                      Enable (unmask) interrupt                                                                                                        |
| 1       | CMPERR   | Compare Error Interrupt Enable.<br>The CRC_INEN.CMPERR enables (unmasks) the data compare interrupt, which is generated when CRC data comparison fails. |
|         |          | 0                      Disable (mask) interrupt                                                                                                         |
|         |          | 1                      Enable (unmask) interrupt                                                                                                        |

Interrupt Enable Set Register

The CRC\_INEN\_SET permits setting individual bits in the CRC\_INEN register without affecting other bits in the register.

CRC\_INEN\_SET: Interrupt Enable Set Register - R/WA

Reset = 0x0000 0000

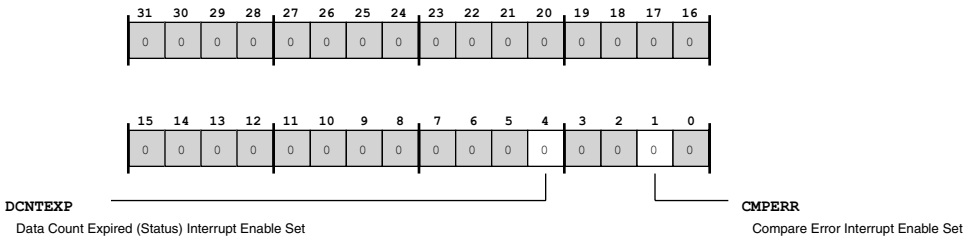


Figure 12-10: CRC\_INEN\_SET Register Diagram

Table 12-14: CRC\_INEN\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                           |           |
|---------|----------|---------------------------------------------------|-----------|
| 4       | DCNTEXP  | Data Count Expired (Status) Interrupt Enable Set. |           |
|         |          | 0                                                 | No Effect |
|         |          | 1                                                 | Set Bit   |
| 1       | CMPERR   | Compare Error Interrupt Enable Set.               |           |
|         |          | 0                                                 | No Effect |
|         |          | 1                                                 | Set Bit   |

Interrupt Enable Clear Register

The CRC\_INEN\_CLR permits clearing individual bits in the CRC\_INEN register without affecting other bits in the register.

CRC\_INEN\_CLR: Interrupt Enable Clear Register - R/W

Reset = 0x0000 0000

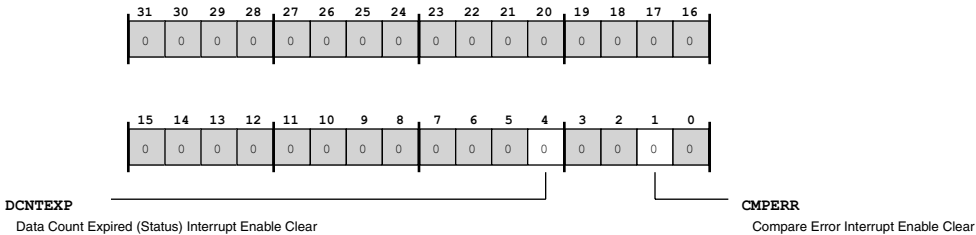


Figure 12-11: CRC\_INEN\_CLR Register Diagram

Table 12-15: CRC\_INEN\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                             |           |
|---------|----------|-----------------------------------------------------|-----------|
| 4       | DCNTEXP  | Data Count Expired (Status) Interrupt Enable Clear. |           |
|         |          | 0                                                   | No Effect |
|         |          | 1                                                   | Clear Bit |

**Table 12-15:** CRC\_INEN\_CLR Register Fields (Continued)

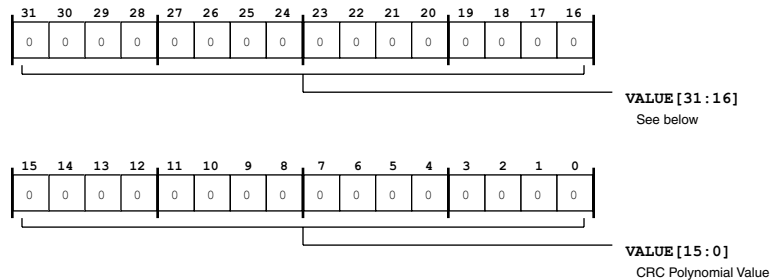
| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 1       | CMPERR   | Compare Error Interrupt Enable Clear. |
|         |          | 0 No Effect                           |
|         |          | 1 Clear Bit                           |

## Polynomial Register

The CRC\_POLY holds a 32-bit polynomial for CRC operations. Bit 31 corresponds to coefficient of  $x^{31}$  of the CRC polynomial, bit 30 corresponds to coefficient of  $x^{30}$ , and so on through to bit 0. Coefficient of  $x^{32}$  is assumed to be "1" for any polynomial that is selected. Based on the polynomial in CRC\_POLY, the CRC generates a look-up table (LUT), which is used to compute the CRC of the incoming data stream.

### CRC\_POLY: Polynomial Register - R/W

Reset = 0x0000 0000

**Figure 12-12:** CRC\_POLY Register Diagram**Table 12-16:** CRC\_POLY Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | CRC Polynomial Value.   |

## Status Register

The CRC\_STAT indicates status for CRC operations and interrupt generation.

CRC\_STAT: Status Register - R/WA

Reset = 0x0000 0000

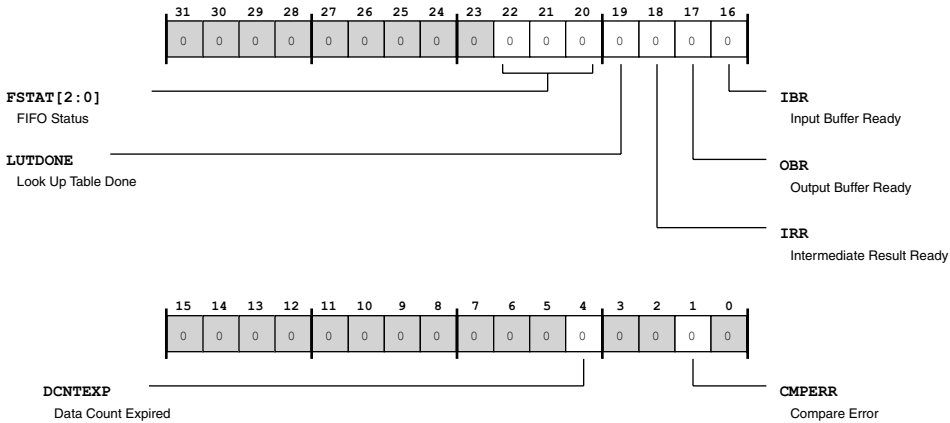


Figure 12-13: CRC\_STAT Register Diagram

Table 12-17: CRC\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22:20   | FSTAT    | FIFO Status.<br>The CRC_STAT.FSTAT indicates the current FIFO status. This field is read-only.                                                                                                                  |
|         |          | 0 FIFO Empty                                                                                                                                                                                                    |
|         |          | 1 FIFO has 1 data                                                                                                                                                                                               |
|         |          | 2 FIFO has 2 data                                                                                                                                                                                               |
|         |          | 3 FIFO has 3 data                                                                                                                                                                                               |
|         |          | 4 FIFO has 4 data (Full)                                                                                                                                                                                        |
| 19      | LUTDONE  | Look Up Table Done.<br>The CRC_STAT.LUTDONE indicates that the CRC has generated the look up table for the current polynomial. This read-only bit is cleared at reset and cleared when the CRC_POLY is written. |
|         |          | 0 No Status                                                                                                                                                                                                     |
|         |          | 1 LUT Generation Done                                                                                                                                                                                           |



**Table 12-17:** CRC\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | IRR      | Intermediate Result Ready.<br>The CRC_STAT.IRR indicates that the CRC has updated the CRC_RESULT_CUR register with intermediate CRC results for the new data written to the CRC. The processor core should read from the CRC_RESULT_CUR register only after detecting CRC_STAT.IRR =1. This read-only bit is cleared by CRC hardware and is valid when CRC_CTL.IRRSTALL is enabled. |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1                      Intermediate Results Ready                                                                                                                                                                                                                                                                                                                                   |
| 17      | OBR      | Output Buffer Ready.<br>The CRC_STAT.OBR indicates that the CRC has data ready for the processor core to read. The processor core should read from the CRC only after detecting CRC_STAT.OBR =1. This read-only bit is cleared by CRC hardware.                                                                                                                                     |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1                      Output Buffer Ready                                                                                                                                                                                                                                                                                                                                          |
| 16      | IBR      | Input Buffer Ready.<br>The CRC_STAT.IBR indicates that the CRC is ready to accept a processor core write. The processor core should write to the input register only after detecting that CRC_STAT.IBR =1. This read-only bit is cleared by CRC hardware.                                                                                                                           |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1                      Input Buffer Ready                                                                                                                                                                                                                                                                                                                                           |
| 4       | DCNTEXP  | Data Count Expired.<br>The CRC_STAT.DCNTEXP indicates that the CRC_DCNT has expired. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on CRC_DCNT expiry, the CRC generates the CRC_INEN.DCNTEXP interrupt.                                                                                                        |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1                      Data Counter Expired                                                                                                                                                                                                                                                                                                                                         |

**Table 12-17:** CRC\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | CMPERR   | Compare Error.<br>The CRC_STAT.CMPERR indicates that a CRC mismatch or data mismatch has been detected. This W1C bit is not automatically cleared when the CRC is disabled (CRC_CTL.BLKEN =0). When the CRC sets this bit on detecting a mismatch, the CRC generates the CRC_INEN.CMPERR interrupt. While this bit is set, the CRC_DCNTCAP is disabled from capturing the data count values. |
|         |          | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Compare Error                                                                                                                                                                                                                                                                                                                                                                              |

## Data Count Capture Register

The CRC\_DCNTCAP captures the CRC\_DCNT value when a compare operation fails in data verify mode. This capture can be used to track the position of error in the data stream. Capture operation is enabled only if the CRC\_STAT.CMPERR indicates no compare error. After an error occurs and data count is captured, no further errors are logged until the CRC\_STAT.CMPERR bit is cleared. To obtain the position of error in the data stream, subtract the CRC\_DCNTCAP value from the initial CRC\_DCNT.

### CRC\_DCNTCAP: Data Count Capture Register - R/W

Reset = 0x0000 0000

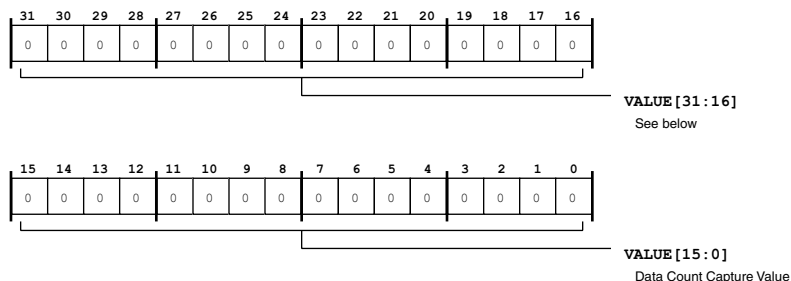


**Figure 12-14:** CRC\_DCNTCAP Register Diagram

**Table 12-18:** CRC\_DCNTCAP Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | VALUE    | Data Count Capture Value. |

## CRC Final Result Register

The CRC\_RESULT\_FIN holds the final CRC computed for a data stream. A data stream is a DMA of CRC\_DCNT number of words into the CRC. When CRC\_DCNT decrements to zero for each datastream, the CRC loads CRC\_RESULT\_FIN with the value from CRC\_RESULT\_CUR.

CRC\_RESULT\_FIN: CRC Final Result Register - R/W

Reset = 0x0000 0000

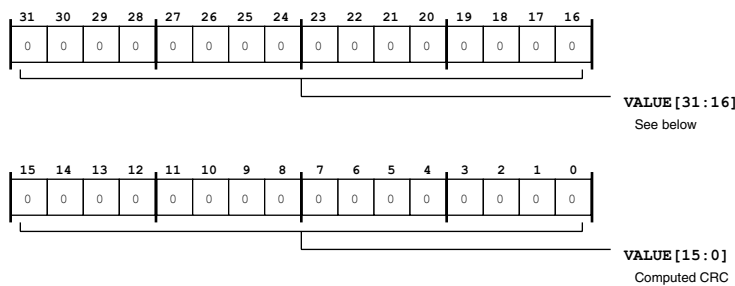


Figure 12-15: CRC\_RESULT\_FIN Register Diagram

Table 12-19: CRC\_RESULT\_FIN Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Computed CRC.           |

## CRC Current Result Register

The CRC\_RESULT\_CUR holds the current or intermediate CRC result and is updated when new data is written into the CRC. Each time the CRC\_DCNT expires, the CRC loads the value from this register into the CRC\_RESULT\_FIN. The CRC\_RESULT\_CUR may be set to auto clear to zero or auto clear to ones when CRC\_DCNT expires by configuring the CRC\_CTL.AUTOCLRZ and CRC\_CTL.AUTOCLR bits. Before starting a CRC operation, the CRC\_RESULT\_CUR should be programmed to the desired value. Note that this register can be read by the processor core at any time.

CRC\_RESULT\_CUR: CRC Current Result Register - R/W

Reset = 0x0000 0000

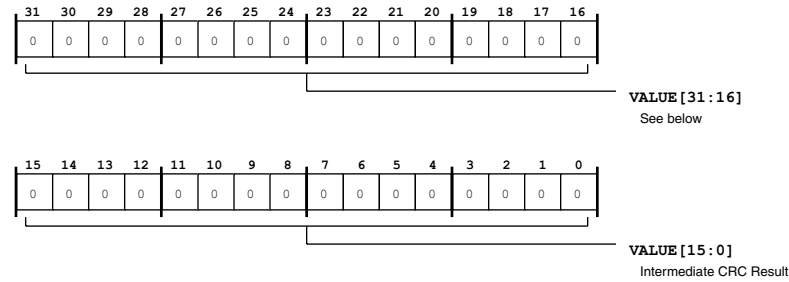


Figure 12-16: CRC\_RESULT\_CUR Register Diagram

Table 12-20: CRC\_RESULT\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Intermediate CRC Result. |

Revision ID Register

The CRC\_REVID contains the CRC module revision information.

CRC\_REVID: Revision ID Register - R/NW

Reset = 0x0000 0000

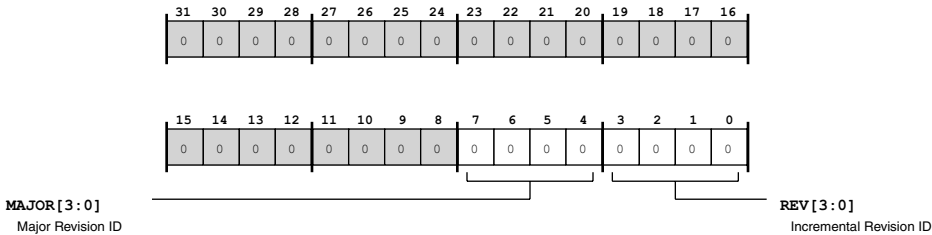


Figure 12-17: CRC\_REVID Register Diagram

Table 12-21: CRC\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 7:4     | MAJOR    | Major Revision ID.       |
| 3:0     | REV      | Incremental Revision ID. |





# 13 Direct Memory Access (DMA)

The DMA channels are dispersed throughout the infrastructure and may be clustered together via system crossbars (SCB) so as to share a single interface with the main system crossbar.

The DMA channels can perform transfers between memory and a peripheral or between one memory and another memory. Two DMA channels are required for memory to memory DMA transfers (MDMA). One channel is the source channel, and the second, the destination channel.

All DMA channels can transport data to and from virtually all on-chip and off-chip memories.

DMA transfers on the processor can be descriptor-based or register-based. Register-based DMA allows the processor to directly program DMA controller registers to initiate a DMA transfer. On completion, the controller registers may be automatically updated with their original setup values for continuous transfer, if needed. Descriptor-based DMA transfers require a set of parameters stored within memory to initiate a DMA sequence. Descriptor-based transfers allow the chaining together of multiple DMA sequences. In Descriptor-based DMA operations, a DMA channel can be programmed to automatically set up and start another DMA transfer after the current sequence completes.

The DMA channel does not connect external memories and devices directly. Rather, data is passed through an external memory interface port. Any kind of device that is supported by the external memory interface can also be accessed by DMA operation. This is typically flash memory, SRAM, DDR SDRAM, FIFOs, or memory-mapped peripheral devices.

## DMA Channel Features

The processor uses Direct Memory Access (DMA) to transfer data within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure and interface with the SCB crossbar.

The following list provides DMA interface features.

- Supports integer byte strides including byte strides of 0 and negative byte strides
- Register based configuration
  - Core writes DMA configuration
  - Supports automatic reloading for continuous operation
- Flexible descriptor based configuration

- DMA descriptors are fetched from memory
  - Support for variable descriptor sizes
- Flexible flow control
  - Transitions between the various descriptor based modes and for DMA termination
- Orthogonal transfers
  - Support for three transfer dimensions
  - 1-D and 2-D transfers supported per descriptor set
  - 3-D support provided by chained descriptor sets
- Configurable memory and peripheral transfer word sizes
  - Memory interface supports 8, 16, 32, 64, 128 and 256 bit transfers
  - Peripheral interface supports for 8, 16, and 32-bit transfers
- Interrupt notification
  - Row or work unit completion
  - Error conditions
- Incoming and outgoing trigger support
  - Trigger generation for row or work unit completion
  - Work unit can wait for incoming trigger
- MMR Access Bus
  - Provides access to memory mapped registers for configuration, monitoring and debug
- SCB Crossbar Interface
  - Connects the DMA Channel to the System Crossbar
- Peripheral DMA Bus
  - Interfaces the DMA channel to a peripheral or another DMA channel
- Peripheral data request interrupt support
- Bandwidth monitoring and limiting

## **DMA Channel Functional Description**

This section provides a functional description of the DMA channel interface.



## DMA Register List

The DMA channel (DMA) supports data transfers within memory spaces or between a memory space and a peripheral. The processor can specify data transfer operations and return to normal processing while the fully integrated DMA channel carries out the data transfers independent of processor activity. The DMA channels are dispersed throughout the infrastructure, as DMAs. A set of registers govern DMA operations. For more information on DMA functionality, see the DMA register descriptions.

**Table 13-1:** DMA Module Registers

| Name           | Description                              |
|----------------|------------------------------------------|
| DMA_DSCPTR_NXT | Pointer to Next Initial Descriptor       |
| DMA_ADDRSTART  | Start Address of Current Buffer          |
| DMA_CFG        | Configuration Register                   |
| DMA_XCNT       | Inner Loop Count Start Value             |
| DMA_XMOD       | Inner Loop Address Increment             |
| DMA_YCNT       | Outer Loop Count Start Value (2D only)   |
| DMA_YMOD       | Outer Loop Address Increment (2D only)   |
| DMA_DSCPTR_CUR | Current Descriptor Pointer               |
| DMA_DSCPTR_PRV | Previous Initial Descriptor Pointer      |
| DMA_ADDR_CUR   | Current Address                          |
| DMA_STAT       | Status Register                          |
| DMA_XCNT_CUR   | Current Count(1D) or intra-row XCNT (2D) |
| DMA_YCNT_CUR   | Current Row Count (2D only)              |
| DMA_BWLCNT     | Bandwidth Limit Count                    |
| DMA_BWLCNT_CUR | Bandwidth Limit Count Current            |
| DMA_BWMCNT     | Bandwidth Monitor Count                  |
| DMA_BWMCNT_CUR | Bandwidth Monitor Count Current          |

## **DMA Definitions**

To make the best use of the DMA channel, it is useful to understand the following terms.

### **Descriptor**

An individual configuration fetched from memory that maps to a single register within a DMA channel.

### **Descriptor Fetch**

The action of retrieving descriptors from memory through memory read operations and loading then into the DMA channel registers upon their read return.

### **Descriptor Set**

A group of descriptors associated with a single work unit.

### **Disabled State**

The channel is disabled because the enable bit (EN) = 0 or as a result of an error.

### **DMAC**

An acronym used for a DMA cluster

### **DMA Channel**

A single DMA engine that has all the capabilities and registers as defined for a given processor. A DMA channel or engine is connected to a single peripheral.

### **DMA Cluster**

A grouping of multiple DMA channels with a shared SCB crossbar interface, controller and arbiter. Also known as a DMAC

### **Initial Descriptor**

The first descriptor in the descriptor set.

### **MDMA**

Memory-to-Memory DMA Data transfer. Two DMA channels are paired to perform a Memory Read from one address location and a Memory Write of that data to another address location.

### **Stop State**

A time where the channel is enabled but not currently programmed to perform a data transfer. Programming the flow to STOP causes the channel to enter Stop State at the end of the work unit.

### **User**

Any person, debug, emulator, software routine or action taken by the Core that accesses the MMR Registers of the DMA channel or peripherals, or sets up data and descriptors in memory.

### **Wait State**

If instructed to wait for a trigger, the channel enters this state once it has completed a work unit. The channel remains in this state until a trigger occurs. If a trigger came in before reaching the wait state, the channel will skip over the Wait State upon completion of the work unit.

### **Work Unit**

A single data transaction or series of data transactions performed based on the configuration of the DMA channel. In the case of autobuffer mode, a new work unit is defined at the time all current count registers are initialized to start values. Once all the current count registers count down to zero, the work unit has completed.

### **Work Unit Chain**

A single work unit or a series of work units separated by a stop or disabled state. The work units in the chain are programmed to another descriptor flow. The last work unit in the chain is programmed to a flow of STOP or AUTO. STOP stops the state at the end of that work unit. AUTO is required to be disabled by disabling the DMA channel. A work unit chain is also known as a descriptor chain.

## **DMA Channel Block Diagram**

The figure shows the functional blocks within the DMA interface.

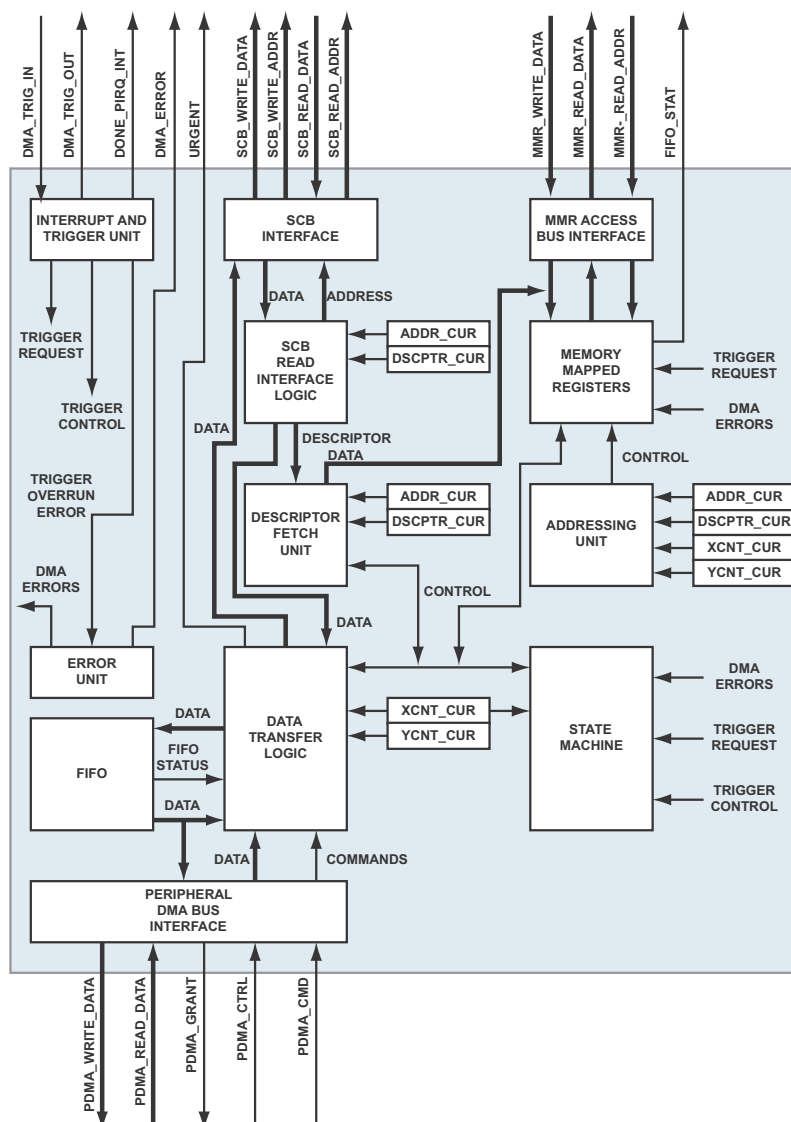


**Figure 13-1: DMA Channel Block Diagram**

For more information on the interfaces shown in the block diagram, see:

- [DMA Channel Peripheral DMA Bus](#)
- [DMA Channel MMR Access Bus](#)
- [DMA Channel Event Control](#)
- [DMA Channel SCB Interface](#)
- [DMA Channel List for ADSP-BF60x](#)

## DMA Channel Architectural Concepts

The DMA channel provides a means to transfer data between memory spaces or between memory and a peripheral via a number of system interfaces. The DMA channel provides an efficient means of distributing data throughout the system freeing up the processor core for other operations. Each peripheral that supports DMA transfers has its own dedicated DMA channel or channels with its own register set that configures and controls the operating modes of the DMA transfers.

### DMA Channel SCB Interface

The SCB interface connects the DMA channel to the SCB crossbar allowing for transfers to and from the processors internal memory and other suitable system resources.

The DMA channel connects to the system interconnect through the SCB interface allowing the DMA channel to perform work unit data transfers with memories such as L1, internal L2 and external L3. In addition to work unit data transfers, the SCB interface is also used for the fetching of descriptor sets for all the descriptor based transfer modes.

The DMA channel is capable of supporting data bus widths of 16, 32, 64 or 128-bits. The data bus widths for a given DMA channel on particular processor may vary and are not configurable. The assigned bus widths can be determined by the user through reading the `DMA_STAT.MBWID` field.

The DMA channel operates at `SCLK` frequency as does the SCB interface. The SCB crossbar handles the internal arbitration of the transfer requests of all the masters interfaced to the SCB crossbar instance.

### SCB Interface Signals

**Table 13-2:** SCB Interface Signals

| Signal            | Width (bits) | Description                                                                                                 |
|-------------------|--------------|-------------------------------------------------------------------------------------------------------------|
| SCB_WRITE_DATA    | 16/32/64/128 | Data bus used for write operations. The width of the bus can be determined from <code>DMA_STAT.MBWID</code> |
| SCB_WRITE_ADDRESS | 32           | Write address bus. Provides the address of the first transfer in a burst transaction                        |
| SCB_READ_DATA     | 16/32/64/128 | Data bus used for read operations. The width of the bus can be determined from <code>DMA_STAT.MBWID</code>  |
| SCB_READ_ADDRESS  | 32           | Read address bus. Provides the address of the first transfer in a burst transaction                         |

## SCB Burst Transfers

The SCB interface supports burst transfers for memory read and write operations. The burst length is a function of the DMA channels configurable memory size for the work unit and the fixed bus width of the DMA channels SCB data bus. If the DMA channel is configured such that the memory transfer size is less than or equal to the DMA channels bus width then the burst length is always 1. If the configured memory size is greater than the SCB interface bus width then the burst length is sufficient to transfer a transaction as specified by the configured memory size.

**Table 13-3:** DMA Channel SCB Burst Lengths

| Configured Memory Size | Burst Length      |                   |                   |                    |
|------------------------|-------------------|-------------------|-------------------|--------------------|
|                        | <i>16-bit Bus</i> | <i>32-bit Bus</i> | <i>64-Bit Bus</i> | <i>128-bit Bus</i> |
| 1 Bytes                | 1                 | 1                 | 1                 | 1                  |
| 2 Bytes                | 1                 | 1                 | 1                 | 1                  |
| 4 Bytes                | 2                 | 1                 | 1                 | 1                  |
| 8 Bytes                | 4                 | 2                 | 1                 | 1                  |
| 16 Bytes               | 8                 | 4                 | 2                 | 1                  |
| 32 Bytes               | 16                | 8                 | 4                 | 2                  |

## Data Address Alignment

In order to prevent addressing errors and maximize bandwidth of the SCB interface to the DMA channel, data addresses must be aligned to be a multiple of the programmable memory size of the DMA channels configuration.

There are situations in which entire work units cannot be transferred at the maximum configurable memory size. In this case the entire work unit may be fulfilled by reducing the configured memory size at the expense of bus bandwidth. However, through the use of descriptor sets, the first descriptor set can be configured to transfer data until the larger memory size alignments are met. A second descriptor set with a larger memory size configuration may then be used to transfer a bulk of the data in the work unit. Finally a third descriptor set may be used with a smaller memory size in order to complete any final data transfers that may not meet the alignment requirements of the previous descriptor set configuration.

**Table 13-4:** DMA Channel Address Alignment Requirements

| Configured Memory Size | Address Restriction |
|------------------------|---------------------|
| 1 Byte                 | No restriction      |

**Table 13-4:** DMA Channel Address Alignment Requirements (Continued)

| Configured Memory Size | Address Restriction |
|------------------------|---------------------|
| 2 Bytes                | ADDR[0] == 0        |
| 4 Bytes                | ADDR[1:0] == 0      |
| 8 Bytes                | ADDR[2:0] == 0      |
| 16 Bytes               | ADDR[3:0] == 0      |
| 32 Bytes               | ADDR[4:0] == 0      |

### Descriptor Set Address Alignment

All descriptor set addresses and descriptors within a descriptor set must be aligned to a 32-bit address. The memory size of the DMA channels configuration is ignored for descriptor set fetches, which avoids the need to align descriptor sets based on the previous descriptor sets memory width configuration. For descriptor sets containing only a single descriptor the transfer takes place as a single 32-bit transfer. For descriptor sets containing multiple descriptors, each 32-bit descriptor is fetched individually and treated as multiple 32-bit transfers.

## DMA Channel Peripheral DMA Bus

The peripheral DMA bus connects the DMA channel to a peripheral or another DMA channel.

### Peripheral DMA Bus

The DMA channel connects to peripherals or other DMA channels via the Peripheral DMA bus. This is a dedicated point-to-point interface supporting data bus widths of 8, 16, 32 or 64 bits. The data bus widths for a given DMA channel on a particular processor may vary and are not configurable. The assigned bus width can be determined by the user through reading the `DMA_STAT.PBWID` field.

The DMA channel operates at SCLK frequency as does the peripheral DMA bus.

### Peripheral DMA Bus Signals

**Table 13-5:** Peripheral DMA Bus Signals

| Signal          | Width (bits) | Description                                                                                                 |
|-----------------|--------------|-------------------------------------------------------------------------------------------------------------|
| PDMA_WRITE_DATA | 8/16/32/64   | Data bus used for write operations. The width of the bus can be determined from <code>DMA_STAT.PBWID</code> |
| PDMA_READ_DATA  | 8/16/32/64   | Data bus used for read operations. The width of the bus can be determined from <code>DMA_STAT.PBWID</code>  |

**Table 13-5:** Peripheral DMA Bus Signals (Continued)

| Signal         | Width (bits) | Description                                                                                                                                                                                      |
|----------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PDMA_DMA_GRANT |              | Control signals to indicate that data is valid for DMA channel read operations (peripheral transmit) and that the DMA channel is ready to receive data for write operations (peripheral receive) |
| PDMA_CMD       | 3            | Used by the peripheral for issuing DMA channel control commands                                                                                                                                  |
| PDMA_CTRL      |              | The control signals used by the peripheral to send various commands to the DMA channel and control the direction of flow                                                                         |

**Table 13-6:** PDMA\_CMD Peripheral DMA Control Commands

| Command | Name                | Description                                                      |
|---------|---------------------|------------------------------------------------------------------|
| b#000   | NOP                 | No operation                                                     |
| b#001   | Restart             | Restarts the current work unit from the beginning                |
| b#010   | Finish              | Finishes the current work unit and starts the next               |
| b#011   | Interrupt           | Immediately sets the DMA completion interrupt in the DMA channel |
| b#100   | Request Data        | Typical DMA data request                                         |
| b#101   | Request Data Urgent | Urgent DMA data request                                          |
| b#110   | Reserved            | Reserved                                                         |
| b#111   | Reserved            | Reserved                                                         |

## DMA Channel Peripheral Control Commands

The Peripheral DMA bus of the DMA channel provides a means for peripherals on the processor to issue commands to the DMA channel to provide greater control over the DMA channel operation allowing for improved real-time performance and relieving control and interrupt demands on the core. Peripherals may send commands to the DMA controller over the 3-bit PERI\_CMD bus. The DMA control commands extend the set of operations available to the peripheral beyond the simple “request data” command used by peripherals in general. Refer to the appropriate peripheral chapter for a description on how that peripheral uses DMA control commands.



While these DMA control commands are not visible to or controlled by the user, their use by a peripheral has implications for the structure of the DMA transfers which that peripheral can support. It is important that application software be written to comply with certain restrictions regarding work units and descriptor chains so that the peripheral operates properly whenever it issues DMA control commands.

### **Idle Command**

This command is driven by the when the DMA channel and the peripheral are enabled and no data requests are required.

### **Restart Command**

This command causes the current work unit to interrupt processing and start again, using the addresses and count values from the DMA\_ADDRSTART, DMA\_XCNT and DMA\_YCNT registers. No interrupt is signalled when the work unit terminates.

If a channel programmed for transmit (memory read) receives a Restart command, the channel momentarily pauses while any pending memory reads initiated prior to the Restart command are completed. During this period of time, the channel does not grant DMA requests. Once all pending reads have been flushed from the channel's pipelines, the channel resets its counters and FIFO, and starts pre fetch reads from memory. DMA data requests from the peripheral are granted as soon as new pre fetched data is available in the DMA FIFO. In this case the peripheral can use the Restart command to reattempt a failed transmission of a work unit.

If a channel programmed for receive (memory write) receives a Restart command, the channel stops writing to memory, discards any data held in its DMA FIFO, and resets its counters and FIFO. As soon as this initialization is complete, the channel again grants DMA write requests from the peripheral. In this case the peripheral can use the Restart command to abort the transfer of received data into a work unit, and reuse the memory buffer for a later data transfer.

The Restart Control Command request is not granted/acknowledged. The request is always accepted by the DMA controller

### **Finish Command**

The Finish Command causes the current work unit to terminate processing and move on to the next work unit. An interrupt/trigger event is signalled as usual, (if enabled within the DMA\_CFG register). The peripheral can then use the Finish command to partition the DMA stream into work units on its own, perhaps as a result of parsing the data currently passing through its supported communication channel, without direct real-time control by the processor.

If a channel is programmed for transmit (memory read) operation and receives a Finish command, the channel momentarily pauses while any pending memory reads initiated prior to the Finish command are completed. During this period of time, the channel does not grant DMA requests. Once all pending reads have been flushed from the channel's pipelines, the channel signals an interrupt/trigger (if enabled), and begins fetching the next descriptor (if any). DMA data requests from the peripheral are granted as soon as new pre fetched data is available in the DMA FIFO.

If a channel programmed for receive (memory write) receives a Finish command, the channel stops granting new DMA requests while it drains its FIFO. Any DMA data received by the DMA channel prior to the finish Command is written to memory. When the FIFO reaches an empty state, the channel signals an interrupt/trigger (if enabled) and begins fetching the next descriptor (if any). Once the next descriptor has been fetched, the channel initializes its FIFO, and then resumes granting DMA requests from the peripheral.

The Finish command request is not granted/acknowledged. The request is always accepted by the DMA channel

## **Interrupt Command**

The Interrupt command causes the DMA channel to generate an interrupt. When programming the channel to support this command, the `DMA_CFG.INT` bit field must be configured to PIRQ mode so that the channel does not generate interrupts based on work unit state, but instead generates interrupts only when it receives the Interrupt command from the peripheral. Upon receiving an Interrupt command, the event is indicated in the `DMA_STAT.PIRQ` bit if all of the following conditions are satisfied.

- The DMA channel is enabled as per the `DMA_CFG.EN` bit.
- The DMA channel is in the stop state.
- The interrupt in `DMA_CFG.INT` is configured for PIRQ mode.

The peripheral only issues the Interrupt command in response to the Last Grant command being received from the DMA channel which indicates that the transfer is the last transfer in the work unit.

## **Request Data Command**

The Request Data command is a request for data transfers between the DMA channel and the peripheral. The request is held by the peripheral until granted/acknowledged by the DMA channel.

## **Request Data Urgent Command**

The Request Data Urgent command behaves identically to the Request Data command, except that while it is asserted the DMA channel performs its memory accesses with urgent priority. This includes both data and descriptor fetch memory accesses. A DMA management capable peripheral might use this control command if, for example, an internal FIFO is approaching a critical condition.

The request is held by the peripheral until granted/acknowledged by the DMA channel

## **DMA Channel Peripheral Control Command Restrictions**

The proper operation of the DMA channel FIFO leads to certain restrictions in the sequence of DMA Peripheral Control commands issued by a peripheral.

## Transmit Restart or Finish

No restart or finish control command may be issued by a peripheral to a channel configured for memory read unless all the following conditions are met:

- The peripheral has already performed at least one DMA transfer in the current work unit.
- The current work unit has  $(FIFO\_SIZE/DMA\_CFG.MSIZE) + 1$  memory transfers remaining.

The first item ensures that the work unit has started. The second item ensures that the work unit has not completed. The second item is sufficiently large that it is always at least five more than the maximum data count prior to any restart or finish command. This implies that any work unit which might be managed by restart or finish commands must have `DMA_XCNT_CUR` and `DMA_YCNT_CUR` register values representing at least five data items.

The second item can be satisfied by ensuring that the number of memory transfers described by the descriptor is  $(FIFO\_SIZE/DMA\_CFG.MSIZE) + 1$  larger than the maximum number of memory transfers expected.

## Receive Restart or Finish

No restart or finish control command may be issued by a peripheral to a channel configured for memory write unless either of the following conditions is met.

- The number of peripheral transfers completed is less than  $(DMA\_CFG.MSIZE/DMA\_CFG.PSIZE) \times$  (Transfers described by descriptor)

In addition to either of the above two conditions, one of the following two conditions must also be met.

- The previous work unit was terminated by a finish command AND the peripheral has done at least one transfer in the current work unit.
- The peripheral has done  $(FIFO\_SIZE/DMA\_CFG.PSIZE) + 1$  transfers in the current work unit.

The first set of conditions ensures that the descriptor is still active while the second set ensures that data from the previous descriptor has left the FIFO and that the current descriptor has started.

## Finish Only

The peripheral has completed exactly  $(DMA\_CFG.MSIZE/DMA\_CFG.PSIZE) \times$  (Transfers described by descriptor) and gives the Restart/Finish command immediately in the next cycle following the last data transfer.

## DMA Channel MMR Access Bus

Provides access to the DMA channels memory mapped registers.

## MMR Access Bus

The MMR access bus provides access to all the DMA channels memory mapped registers for DMA channel configuration, monitoring and debug. The interface has a fixed 32-bit data bus for read and write accesses.

## MMR Access Bus Signals

The following table provides descriptions of the MMR access bus signals.

**Table 13-7:** MMR Access Bus Signals

| Signal         | Width (bits) | Description                                                   |
|----------------|--------------|---------------------------------------------------------------|
| MMR_WRITE_DATA | 32           | Data bus used for write operations to the MMRs from the core. |
| MMR_READ_DATA  | 32           | Data bus used to return read data from the MMRs               |
| MMR_READ_ADDR  | 7            | Address that is used to select the MMR to access              |

## DMA Channel Operation Flow

The flow of operation of the DMA channel is described in the following topics:

- [DMA Channel Startup](#)
- [DMA Channel Refresh](#)
- [DMA Operating Modes](#)
- [DMA Channel Stop Mode](#)
- [DMA Channel Errors](#)

## DMA Channel Startup

In order to enable a DMA operation on a given channel, some or all of the DMA parameter registers must first be written directly. The minimum set of register required to be initialized is dependent upon the desired mode of operation.

## DMA Channel Minimum Enable Requirements

To start a DMA operation on a given channel, some or all of the DMA parameter registers must first be initialized corresponding to the DMA channels desired operating mode

- For descriptor array based flow modes.
  - At a minimum the DMA\_DSCPTR\_CUR register must be written prior to writing to the DMA\_CFG register, which is the special action required to start the DMA channel.

- For descriptor list based flow modes.
  - At a minimum the `DMA_DSCPTR_NXT` register must be written prior to writing to the `DMA_CFG` register, which is the special action required to start the DMA channel.
- For non descriptor based flow modes, the `DMA_STARTADDR`, `DMA_XCNT` and `DMA_XMOD` registers must be written prior to the `DMA_CFG` register.

Programs can write other registers that might remain static throughout the course of the DMA activity. The DMA operation begins once the `DMA_CFG` register is written.

**ATTENTION:** When the `DMA_CFG` register is written directly by software the DMA controller recognizes this as the special startup condition that occurs when starting DMA for the first time on this channel or after the DMA channel is stopped. It is possible for a DMA error condition to be flagged regardless of the `DMA_CFG.EN` bit setting.

## DMA Channel Startup Operation

When the `DMA_CFG` register is written directly by software, the DMA channel recognizes this as the special startup condition that occurs when starting DMA for the first time on this channel or after the channel has entered to the Stop state.

When the descriptor fetch is complete and the DMA channel is enabled, the `DMA_CFG` descriptor element that was read into the `DMA_CFG` register assumes control. Before this point, the direct write to the `DMA_CFG` register had control.

At startup the selected flow mode and the and descriptor size determine the course of the DMA initialization process. The `DMA_CFG.FLOW` field determines whether to load more current registers from descriptor sets in memory, while `DMA_CFG.NDSIZE` details how many descriptor elements to fetch before starting the DMA. DMA registers not included in the descriptor are not modified from their prior values

For descriptor list flow modes the `DMA_DESC_PTR_NXT` register is copied into the `DMA_DESC_PTR_CUR` register. Then, fetches of new descriptor elements from memory are performed, indexed by the `DMA_DESC_PTR_CUR` register, which is incremented after each fetch. After completion of the descriptor fetch, the `DMA_DESC_PTR_CUR` register points to the next 32-bit word in memory past the end of the descriptor.

If the descriptor fetch is for a descriptor array mode transfer then the `DMA_DESC_PTR_NXT` register is not copied into the `DMA_DESC_PTR_CUR` register. Instead the descriptor fetch indexing begins with the value in the `DMA_DESC_PTR_CUR` register.

If `DMA_CFG` is not part of the fetched descriptor set, then the previous value, (originally as written on startup) control the work unit operation. If the `DMA_CFG` register is part of the fetched descriptor set then the value programmed by the MMR access control only the loading of the first descriptor fetched from memory. The subsequent DMA work units are controlled by the configuration of `DMA_CFG` register of the fetched descriptor set.

Once the descriptor fetch is complete or if the flow was originally configured for one of the register based flow modes then the DMA operation begins. The DMA channel immediately fills its FIFO. For a memory

write operation the DMA channel begins accepting data from its peripheral. In the case of a memory read operation the DMA channel begins memory reads when the DMA channel is granted access to the SCB bus.

When the DMA channel performs its first data memory access, its address and count computations take their input operands from the start registers (DMA\_ADDRSTART, DMA\_XCNT and DMA\_YCNT if required), and writes results back to the current registers (DMA\_ADDR\_CUR, DMA\_XCNT\_CUR and DMA\_YCNT\_CUR). Note also that the current registers are not valid until the first memory access is performed, which may be some time after the channel is started by the write to the DMA\_CFG register. The current registers are loaded automatically from the appropriate descriptor elements, overwriting their previous contents, as follows:

- DMA\_ADDRSTART is copied to DMA\_ADDR\_CUR
- DMA\_XCNT is copied to DMA\_XCNT\_CUR
- DMA\_YCNT is copied to DMA\_YCNT\_CUR

## **DMA Channel Refresh**

Upon completion of a work unit, the DMA channel performs the following operations:

- Completes the transfer of all data between memory and the DMA channel.
- If the DMA channel is configured for a memory read operation with the DMA\_CFG.SYN bit enabled, then a synchronized transition takes place. The DMA channel transfers all data to the peripheral before continuing.
- If interrupts/triggers are enabled, then the signals are forwarded from the DMA channel and the DMA\_STAT register is updated to indicate the interrupt/trigger events.
- If the flow was set to Stop mode, the DMA operation stops by setting the DMA\_STAT.RUN bit field to indicate the channel is no longer running. Any remaining data in the DMA channel's FIFO is transferred to the peripheral.
- For descriptor array mode:
  - Loads a new descriptor from memory into the DMA registers by way of the contents of the DMA\_DESC\_PTR\_CUR register, while incrementing the DMA\_DESC\_PTR\_CUR register. The descriptor size is taken from the DMA\_CFG.NDSIZE value prior to the fetch.
- For descriptor list mode:
  - Copies the DMA\_DESC\_PTR\_NXT register into the DMA\_DESC\_PTR\_CUR register. Next, the DMA channel fetches the descriptor from the new contents of the DMA\_DESC\_PTR\_CUR register and places these contents into the DMA registers while incrementing the DMA\_DESC\_PTR\_CUR register.
- For descriptor on demand array mode:

- Checks to see if an incoming trigger event has been detected.

If a trigger event has been detected, then the DMA channel loads a new descriptor from memory into the DMA registers from the contents of the `DMA_DESC_PTR_CUR` register, while incrementing the `DMA_DESC_PTR_CUR` register. The descriptor size is taken from the `DMA_CFG.NDSIZE` value prior to the fetch.

If a trigger event was not detected then the DMA channel begins the next work unit by reloading the current registers as described below.

- For descriptor on demand list mode:

- Checks to see if an incoming trigger event has been detected.

If a trigger event was detected, then the DMA channel copies the `DMA_DESC_PTR_NXT` register into the `DMA_DESC_PTR_CUR` register. Next, the DMA channel fetches the descriptor memory from the new contents of the `DMA_DESC_PTR_CUR` register and places these contents into the DMA registers while incrementing the `DMA_DESC_PTR_CUR` register.

If a trigger event was not detected then the DMA channel begins the next work unit by reloading the current registers as described in the step below.

- If flow was configured for anything other than Stop mode then the DMA channel begins the next work unit by reloading the current registers (`DMA_ADDR_CUR`, `DMA_XCNT_CUR` and `DMA_YCNT_CUR`) from their descriptor registers (`DMA_ADDRSTART`, `DMA_XCNT` and `DMA_YCNT`).

## DMA Channel Work Unit Transitions

Transitions from one work unit to the next are controlled by `DMA_CFG.SYNC` bit for a given work unit. In general, continuous transitions have lower latency at the cost of restrictions on changes of data format or addressed memory space in the two work units. These latency gains and data restrictions arise from the way the DMA FIFO pipeline is handled while the next descriptor is fetched.

In continuous transitions where synchronization is disabled, the DMA FIFO pipeline continues to transfer data to and from the peripheral or destination memory during the descriptor fetch and/or when the DMA channel is paused between descriptor chains. On the other hand, synchronized transitions provide better real-time synchronization of interrupts and triggers with a given peripheral state. Synchronized transitions also provide greater flexibility in the data formats and memory spaces of the two work units, at the cost of higher latency in the transition. In synchronized transitions, the DMA FIFO pipeline is drained to the destination or flushed (received data discarded) between work units.

**NOTE:** Work unit transitions for MDMA streams are controlled by the `DMA_CFG.SYNC` bit of the MDMA source channel. The `DMA_CFG.SYNC` bit of the MDMA destination channel is reserved and must be set to disabled state. In transmit (memory read) channels, the `DMA_CFG.SYNC` bit of the last descriptor prior to the transition controls the transition behavior. In contrast, in receive channels, the `DMA_CFG.SYNC` bit of the first descriptor of the next descriptor chain controls the transition.

## DMA Channel Transmit and MDMA Source Transitions

In DMA transmit (memory read) and MDMA source channels, the `DMA_CFG.SYNC` bit controls the interrupt timing at the end of the work unit and the handling of the DMA FIFO between the current and next work unit.

If the `DMA_CFG.SYNC` bit is configured to disable synchronization, a continuous transition is selected. In a continuous transition, just after the last data item is read from memory, these four operations all start in parallel:

- The interrupt/trigger (if any) is signalled.
- The `DMA_STAT` register is updated to indicate DMA status is completed.
- The next descriptor begins to be fetched.
- The final data items are delivered from the DMA FIFO to the destination memory or peripheral.

This allows the DMA channel to provide data from the FIFO to the peripheral continuously during the descriptor fetch latency period.

When synchronization is disabled, the final interrupt/trigger (if enabled) occurs when the last data is read from memory. This event occurs at the earliest time that the output memory buffer may safely be modified without affecting the previous data transmission. There may be a number of data items still remaining in the FIFO and not yet at the peripheral. This number is dependent upon the FIFO depth of the DMA channel. Therefore, in this configuration, the DMA interrupt should not be used as the sole means of synchronizing the shutdown or re configuration of the peripheral following a transmission.

**NOTE:** If continuous transition is selected on a transmit (memory read) descriptor, the next descriptor is required to have the same peripheral transfer size (`DMA_CFG.PSIZE`), read/write direction, and source memory (internal versus external) as the current descriptor.

Disabling synchronization, to select continuous transition on a work unit that is configured for Stop flow mode with interrupt/trigger enabled, can result in the event service routine being executed while the final data is still draining from the FIFO to the peripheral. This is indicated by the DMA channels `DMA_STAT.RUN` bits—if the channel is still running then the FIFO is not yet empty. Do not start a new work unit with different peripheral transfer size or direction while the channel is still running. Further, if the channel is disabled via the `DMA_CFG.EN` bit, the data in the FIFO is lost.

A synchronized transition allows the DMA FIFO to first be drained to the destination memory or peripheral before any interrupt is signalled, and before any subsequent descriptor or data is fetched. This incurs greater latency, but provides direct synchronization between the DMA interrupt and the state of the data at the peripheral.

For example, if synchronization is enabled and interrupts are enabled on the last descriptor in a work unit, the interrupt occurs when the final data is transferred to the peripheral. This allows the service routine to properly switch to non-DMA transmit operation. When the interrupt service routine is invoked, the DMA channel FIFO will be empty and the DMA channel will not still be running as indicated by the `DMA_STAT.RUN` bits.



A synchronized transition also allows greater flexibility in the format of the DMA descriptor chain. When enabled, the next descriptor may have any `DMA_CFG.PSIZE` configuration or read/write direction supported by the peripheral and may come from either memory space (internal as opposed to external). This can be useful in managing MDMA work unit queues, since it is no longer necessary to interrupt the queue between dissimilar work units.

## **DMA Channel Work Unit Receive and MDMA Destination Transitions**

In DMA receive (memory write) channels, the `DMA_CFG.SYNC` bit controls the handling of the DMA FIFO between descriptor chains (not individual descriptor sets), when the DMA channel is paused. The DMA channel pauses after descriptor sets configured with Stop flow mode complete, and the channel may be restarted (for example, after an interrupt) by writing the channel's `DMA_CFG` register with a value that enables the DMA channel. If the synchronization is disabled in the new work unit's `DMA_CFG` value, a continuous transition is selected. In this mode, any data items received into the DMA FIFO while the channel was paused are retained, and they are the first items written to memory in the new work unit. This mode of operation provides lower latency at work unit transitions and ensures that no data items are dropped during a DMA pause, at the cost of certain restrictions on the DMA descriptors.

**NOTE:** If the `DMA_CFG.SYNC` bit is configured to disable synchronization on the first descriptor of a descriptor chain after a DMA pause, the `DMA_CFG.PSIZE` field of the new chain must not change from the configuration of the previous descriptor chain that was active before the pause, unless the DMA channel is reset between chains by disabling and then re-enabling the DMA channel.

A synchronized transition is selected if the `DMA_CFG.SYNC` field is configured to enable synchronization. In this mode, only the data received from the peripheral by the DMA channel after the write to the `DMA_CFG` register is delivered to memory. Any prior data items transferred from the peripheral to the DMA FIFO before this register write are discarded. This provides direct synchronization between the data stream received from the peripheral and the timing of the channel restart (when the `DMA_CFG` register is written).

For receive DMA operations, the synchronization has no effect in transitions between work units in the same descriptor chain (that is, when the previous descriptor's flow mode was not STOP, so that DMA channel did not pause).

If a descriptor chain begins with synchronization enabled, there is no restriction on `DMA_CFG.PSIZE` of the new chain in comparison to the previous chain.

**NOTE:** The peripheral transfer size (`DMA_CFG.PSIZE`) must not change between one descriptor and the next in any DMA receive (memory write) channel within a single descriptor chain, regardless of the `SYNC` bit setting. In other words, all memory write descriptor sets in a descriptor chain must have the same `DMA_CFG.PSIZE` value. For any DMA receive (memory write) channel, there is no restriction on changes of peripheral transfer size (internal versus external) between descriptors or descriptor chains.

## **DMA Channel Transfer Termination and Shutdown**

This section describes channel transfer termination and shutdown in stop flow mode and in autobuffer flow mode.

## Stop Flow Mode

In Stop flow mode, the DMA channel stops automatically after the work unit is complete. If a list or array of descriptors is used to control DMA transfers, and if every descriptor contains a `DMA_CFG` descriptor element, then the final `DMA_CFG` descriptor element should have the flow configured to Stop mode setting to gracefully stop the channel. Upon completion the DMA channel remains in the Stop state. This state should not be confused with the disabled state which occurs either due to a DMA error or by configuring `DMA_CFG.EN` so as to disable the DMA channel.

Disabling the DMA channel via a write to the `DMA_CFG.EN` bit is intended to shut down the DMA channel and enter the disabled state. All memory and peripheral data transfers cease and only peripheral interrupts are passed through the DMA channels interrupt signals. However, the DMA channel maintains the `DMA_STAT.RUN` bits. Therefore, in the case of a memory write operation, the outstanding memory transaction counter keeps track of returning memory write acknowledgements and updates as required.

In the case of memory read operations, the outstanding memory transaction counter also keeps track of returning memory reads. However, the memory reads are not written into the FIFO. The counter is updated to reflect the completion of the transaction, but the data is ignored. The `DMA_STAT.RUN` bits remain in the “Waiting for Write ACK/FIFO Drain to Peripheral” state and do not change to “Stop/Idle State” until all outstanding transactions have returned.

When the DMA channel is enabled again via the `DMA_CFG.EN` bit, a full reset is performed and all counters are cleared. If an outstanding memory transaction returns an acknowledgement or read data after this event, then a memory transaction error has occurred and an error is generated. Programs must be sure that all outstanding memory transactions have been completed before re configuring the DMA channel. One method programs may use is to poll the `DMA_STAT.RUN` bits to return to the “Stop/Idle State” before proceeding.

## Autobuffer Flow Mode

In the case of Autobuffer flow modes, the only way to cease operations is to disable the DMA channel via the `DMA_CFG.EN` bit. Therefore, one method of changing to a new work unit would be to disable the DMA channel, set up all the registers (and descriptors in memory, if used) except for `DMA_CFG`, then poll `DMA_STAT.RUN` to wait for the status to reflect “Stop/Idle State”, and finally write `DMA_CFG` to the new configuration to begin the next work unit.

In autobuffer flow mode, or if a list or array of descriptor sets without `DMA_CFG` descriptors, then the DMA transfer process must be terminated by an MMR write to the `DMA_CFG` register with a value whose `DMA_CFG.EN` bit is configured to disable the DMA channel.

**CAUTION:** Interrupt logic based on work unit transitions are disabled when the DMA channel is disabled. Programmers should be aware of their environment and current actions so that additional interrupts are not required from the DMA channel.

**CAUTION:** The DMA channel completes any transactions that have begun and avoids generating bus errors if disabled through `DMA_CFG.EN` in the middle of a transaction. However, the action of re-enabling the DMA is considered a hard reset for all internal DMA channel components.

Therefore, programmers must pay special attention to that particular action in order to avoid unexpected results.

## **DMA Channel Errors**

When an error occurs, the DMA channel maintains all the state and register values which allows programs to diagnose error causes more thoroughly. The greatest benefit to the programmer is to know exactly what operational state the DMA channel was in at the exact moment the error occurred.

It is the responsibility of the programmer to take special care to ensure the root cause of the error is addressed, whether the problem originated in the DMA channel or not. If not properly resolved, the error could result in an additional error shortly after operations resume. The problem may have caused other errors elsewhere in the DMA channel or associated modules and circuitry, therefore care must be taken to address those potential problems also. Finally, the programmer must ensure that all outstanding memory reads and writes are complete, or cleared, before resuming DMA channel operation.

Once all issues have been addressed and all side effects of any error are neutralized, the programmer may clear the `DMA_STAT.ERR` status field and restart the DMA channel by disabling then re-enabling the DMA channel through the `DMA_CFG.EN` bit.

## **Status and Debug**

DMA channel error conditions can cause the DMA process to end abnormally. DMA error detection is provided as a tool for system development and debug, as a way to detect DMA related programming errors. When the DMA channel detects an error, the channel is immediately stopped and any memory read transactions that are returned are discarded. The DMA channels `DMA_STAT.RUN` field is set to indicate idle state, once all outstanding memory transactions are acknowledged. In addition, an error interrupt is asserted and the `DMA_STAT.IRQERR` is updated to reflect this. Also the error cause of the first detected error is updated in the `DMA_STAT.ERRC` field. Unless the error occurs at the exact moment that register values are being modified, the registers will contain their values.

It is possible for error interrupt signals to be combined. Combined error signals requires that the `DMA_STAT` register of each DMA channel associated with a combined error interrupt be read to determine the DMA channel responsible for the generation of the interrupt.

The DMA channel error interrupt handler is required to perform the following actions:

- Read each DMA channel's `DMA_STAT` register to look for a channel with the `DMA_STAT.IRQERR` set to indicate an error.
- Read the DMA channel's `DMA_STAT.ERRC` field to determine the cause of the error.
- Clear the problem with the DMA channel, for example fix the register values.
- Clear the error in the DMA channel via a write 1 to clear operation to the `DMA_STAT.IRQERR` bit.

If any error other than a bandwidth monitor error is already flagged and is not cleared, no other error is reported. If a bandwidth monitor error was reported and not cleared, any newly detected error would be in the updated `DMA_STAT.ERRC` field.

## DMA\_CFG Register Errors

These errors are only flagged when the DMA channel is enabled via the DMA\_CFG.EN bit.

- Reserved setting was used
- DMA\_CFG.TWAIT enabled in Descriptor On Demand Flow mode
- Illegal DMA\_CFG.NDSIZE
- Illegal DMA\_CFG.MSIZE
- DMA\_CFG.MSIZE exceeds the DMA channel's FIFO size
- Illegal DMA\_CFG.PSIZE
- DMA\_CFG.PSIZE exceeds the FIFO size
- DMA\_CFG.PSIZE exceeds the bus width
- Memory Read (Transmit operation), cannot change to receive unless properly synced in the previous work unit, or if first work unit in a new chain
- Memory Read (Transmit operation), cannot change DMA\_CFG.PSIZE unless properly synced in previous work unit, or if first work unit in a new chain
- Memory Write (Receive operation), cannot change to transmit during a descriptor chain. Can only change from receive to transmit if new transmit is synced and first work unit
- Memory Write (Receive operation), cannot change DMA\_CFG.PSIZE unless first work unit with DMA\_CFG.SYNC enabled

## Illegal Register Write During Run

Write to writable registers when the DMA channel is enabled and running are blocked and generate an error. The DMA\_STAT, DMA\_BWLCNT and DMA\_BWMCNT registers are exempt from this behavior.

## Address Alignment Error

An address alignment error is generated when a descriptor address is not aligned to a 32-bit boundary or a transfer address is not aligned for the current DMA\_CFG.MSIZE configuration.

## Memory Access Error

A memory access error is generated when an attempt was made to access an address not populated, defined as cache, or there was a security violation. This error is triggered by an error returned from the memory.

## Trigger Overrun Error

A trigger overrun error is generated when a new trigger input occurred while an outstanding trigger is waiting. this error is only generated if DMA\_CFG.TOVEN is enabled.

## Bandwidth Monitor Error

This error is generated when the bandwidth monitor count expired. this is not a fatal error and the DMA channel will continue its operation.

## Control Interface Error

Control interface errors are reported as bus errors to the bus master. This can be as a result of:

- An address error
- Write to a read-only register

## DMA Channel Memory DMA

A Memory DMA (MDMA) channel provides a means of doing memory-to-memory DMA transfers among the various memory spaces that have DMA support.

Memory DMA (MDMA) channels are implemented by interfacing two DMA channels via the Peripheral DMA Bus Interface. One DMA channel is used for memory read operations and the second is used for memory writes. Depending on the processor, a memory DMA channel may have an additional peripheral, such as a CRC peripheral, inserted into the peripheral DMA bus that may optionally be enabled.

MDMA channel configurations that do not involve an additional peripheral impose no restrictions on which of the DMA channels is to be used for the read operation and which is to be used for the write operation so long as both are not configured for the same transfer direction. For MDMA channel configurations that enable a peripheral between the read and write channels, restrictions may be imposed on which channel may be used for a given transfer direction.

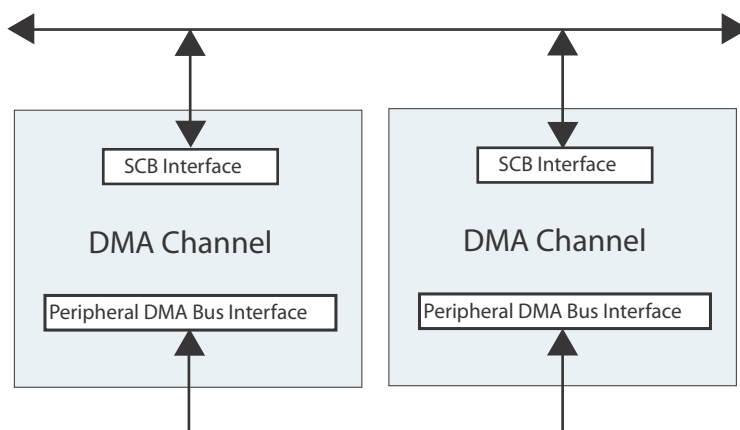


**Figure 13-2:** MDMA Channel Dedicated Pair

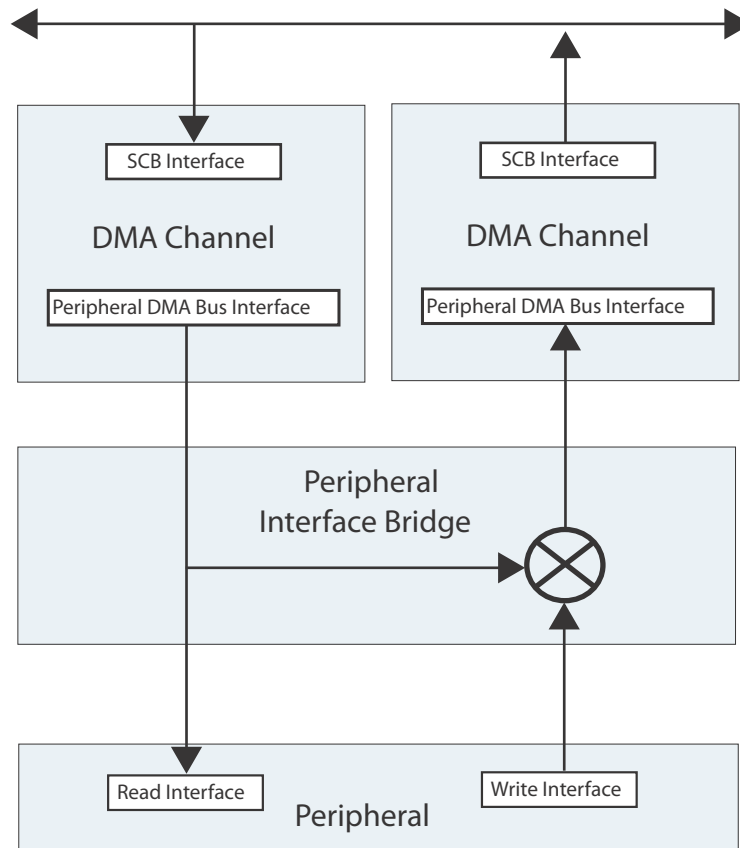


**Figure 13-3: MDMA Channel Pair with Peripheral**

A memory-to-memory transfer always requires the source and destination channels to be enabled. Because the channels are interfaced through the peripheral DMA bus, and because the channel may have an additional peripheral inserted into the peripheral DMA bus, programs must ensure that the `DMA_CFG.PSIZE` of both the source and destination channels are set to the same values.

The memory DMA channels support the full range of `DMA_CFG.MSIZE` options for the DMA transfers to and from the memories.

As the MDMA channel consists of two DMA channels, the entire MDMA channel has two sets of FIFOs, one in the read channel and one in the write channel. This allows for more efficient bursting of both read and write transactions in order to make use of the available bandwidth. While the `DMA_CFG.PSIZE` configuration must be identical for both source and destination DMA channels, this restriction is not imposed for the `DMA_CFG.MSIZE` configuration.

The independent source and destination DMA channels also have their own dedicated interrupt and trigger events, and while it is normal practice to only have event generation performed at destination DMA completion, programs are not restricted to this means of interrupt generation.

Configuration of an MDMA transfer is done in a similar manner to peripheral DMA transfers with the exception of writing two DMA channel registers instead of one.

## MDMA Triggering

To control the pace of data transfers, triggers may be used on either the memory read or the memory write channel pair used in an MDMA operation. Enabling `DMA_CFG.TWAIT` in the memory read channel will prevent both channels from transferring data before the system is ready. However, only configuring the memory write channel to wait for a trigger will allow for data to be fetched from the memory in anticipation of the memory write operation.

## DMA Operating Modes

The DMA channel supports a number of different flow modes that determines how the DMA channel progresses from one work unit to the next.

The flow mode of a DMA channel is not a global setting. A DMA descriptor set may include the descriptor responsible for configuring the flow of the work unit and there is no restriction that the flow must be configured the same for the entire descriptor chain. If the descriptor chain is not endless then the last descriptor set would configure the flow to stop mode resulting in termination of the descriptor chain after work unit completion. Another example for mixing flow modes would be to create an endless descriptor array. The last descriptor set in the array would be configured for list mode and the next descriptor pointer of this descriptor set would point to the first descriptor in the array.

## DMA Channel Register Based Flow Modes

Register-based DMA operations require configuration by directly writing to the DMA channels Memory Mapped Registers registers.

Register-based DMA is the traditional method of DMA operation. Software writes all the DMA channels configuration into the memory-mapped registers. This includes information such as the source or destination address and length of the data to be transferred and then starts the DMA channel operation. The DMA channel supports the following register-based flow modes:

- *DMA Channel Stop Mode*
- *Autobuffer Mode*

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor. The supported descriptor set sizes can differ between the various descriptor based flow modes. In addition to the descriptor set size being configurable, descriptor based DMA also allows for the flow mode of the next descriptor set to be altered allowing for the transition from descriptor array mode to descriptor list mode in addition to configuring the flow to stop or even autobuffer mode.

## DMA Channel Stop Mode

In Stop mode, the DMA operation is executed only once. If started, the DMA channel transfers the desired number of data words and stops itself again when finished. If the DMA channel is no longer used, software configures the enable bit to disable a paused channel. Interrupts and triggers may also be generated for each row/work unit completion depending upon desired configuration.

## Autobuffer Mode

In autobuffer mode, the DMA operates repeatedly in a circular manner. If all data words have been transferred, the address pointer (`DMA_ADDR_CUR`) is reloaded automatically with the `DMA_ADDRSTART` value. An interrupt may also be generated.

Autobuffer mode is enabled via the `DMA_CFG.FLOW` field. The `DMA_CFG.NDSIZE` field must be configured such that the next descriptor size is zero.

## DMA Channel Descriptor Based Flow Modes

Descriptor-based DMA operations fetch descriptor sets from memory allowing for autonomous loading of work units upon another work units.

Software is not required to set up the DMA sequences directly by writing into the DMA controller registers. Rather, software keeps DMA descriptor sets in memory. On demand the DMA controller autonomously loads the descriptor set from memory to the affected DMA controller registers. The descriptor sets can be fetched from any memory space that supports DMA read operations. The descriptor set describes the next operation to be performed by the DMA controller. The descriptor set may include information such as the DMA configuration word as well as data source/destination address, transfer count, and address modify values.

A descriptor set describes a single work unit, however some values from one descriptor set may be reused in the following work unit if they are not overwritten in the subsequent descriptor set fetches and the work unit requires the use of this descriptor.

The DMA channel supports the following flow modes with descriptor based operations:

- *DMA Channel Descriptor Array Mode*
- *DMA Channel Descriptor List Mode*
- *DMA Channel Descriptor On Demand Array Mode*
- *DMA Channel Descriptor On Demand List Mode*

The DMA channel supports variable descriptor set sizes within the configuration. The size of a descriptor set can contain as little as a single descriptor. The supported descriptor set sizes can differ between the various descriptor based flow modes. In addition to the descriptor set size being configurable, descriptor based DMA also allows for the flow mode of the next descriptor set to be altered. This allows programs to



transition from one descriptor based mode to another descriptor based mode in addition to a transition to any of the register-based flow modes.

## DMA Channel Descriptor Array Mode

When configured in this mode, the descriptor sets do not contain further descriptor pointers. The initial descriptor pointer value is written by software and points to an array of descriptors. The individual descriptors are assumed to reside next to each other and, therefore, their address is known.

### Descriptor Sets

The *Parameter Registers and Descriptor Offsets* table illustrates how a descriptor set must be structured in memory. Note that descriptor sets must reside in a contiguous block of memory in the format shown in the table. That is to say that the first descriptor of the next descriptor set must be located in the memory location immediately following the last descriptor of the current descriptor set. The values have the same order as the corresponding MMR offset addresses.

**Table 13-8:** Parameter Registers and Descriptor Offsets

| Descriptor Offset | Parameter Register |
|-------------------|--------------------|
| 0x00              | DMA_ADDRSTART      |
| 0x04              | DMA_CFG            |
| 0x08              | DMA_XCNT           |
| 0x0C              | DMA_XMOD           |
| 0x10              | DMA_YCNT           |
| 0x14              | DMA_YMOD           |

All other DMA channel registers not loaded as a result of the descriptor set fetch retain their previous values. All of the current registers are reloaded between the descriptor set fetch and the start of the DMA operation for the work unit.

### Minimum Startup Requirements

At a minimum the DMA\_DSCPTR\_CUR register must be written prior to writing to the DMA\_CFG register, which is the special action required to start the DMA channel.

## DMA Channel Descriptor List Mode

In this flow mode, multiple descriptors form a chained list in which each descriptor set contains a pointer to the next descriptor set, allowing greater flexibility in memory layout options. When the descriptor set is fetched, this pointer value is loaded into the DMA channels next descriptor pointer register.

### Descriptor Sets

The [Descriptor List Mode Parameter and Descriptor Offsets](#) illustrates how a descriptor set must be structured in memory. Note that while the descriptor sets can be dispersed throughout memory and reside in different memory blocks, each descriptor of the descriptor set must reside in a contiguous section of memory in the format shown in the table. The values have the same order as the corresponding MMR offset addresses.

**Table 13-9:** Descriptor List Mode Parameter and Descriptor Offsets

| Descriptor Offset | Parameter Register |
|-------------------|--------------------|
| 0x00              | DMA_DSCPTR_NXT     |
| 0x04              | DMA_ADDRSTART      |
| 0x08              | DMA_CFG            |
| 0x0C              | DMA_XCNT           |
| 0x10              | DMA_XMOD           |
| 0x14              | DMA_YCNT           |
| 0x18              | DMA_YMOD           |

All other DMA channel registers not loaded as a result of the descriptor set fetch retain their previous values. All of the register's current values are reloaded between the descriptor set fetch and the start of the DMA operation for the work unit.

### Minimum Startup Requirements

At a minimum the DMA\_DSCPTR\_NXT register must be written prior to write to the DMA\_CFG register which is the special action required to start the DMA channel.

## DMA Channel Descriptor On Demand Array Mode

When configured in this mode, the descriptor sets do not contain further descriptor pointers. The initial descriptor pointer value is written by software and points to an array of descriptors. The individual descriptors are assumed to reside next to each other and, therefore, their address is known.

At the end of the work unit, if the DMA channel has not detected an incoming trigger event, the current work unit is repeated. If the DMA channel receives an incoming trigger before completion of the work unit then a new descriptor set is fetched.

## Descriptor Sets

The *Parameter Registers and Descriptor Offsets* table illustrates how a descriptor set must be structured in memory. Note that descriptor sets must reside in a contiguous block of memory in the format shown in the table. That is to say that the first descriptor of the next descriptor set must be located in the memory location immediately following the last descriptor of the current descriptor set. The values have the same order as the corresponding MMR offset addresses.

**Table 13-10:** Parameter Registers and Descriptor Offsets

**Table 13-11:**

| Descriptor Offset | Parameter Register |
|-------------------|--------------------|
| 0x00              | DMA_ADDRSTART      |
| 0x04              | DMA_CFG            |
| 0x08              | DMA_XCNT           |
| 0x0C              | DMA_XMOD           |
| 0x10              | DMA_YCNT           |
| 0x14              | DMA_YMOD           |

All other DMA channel registers not loaded as a result of the descriptor set fetch retain their previous values. All of the current registers are reloaded between the descriptor set fetch and the start of the DMA operation for the work unit.

## Minimum Startup Requirements

At a minimum the DMA\_DSCPTR\_CUR register must be written prior to writing to the DMA\_CFG register, which is the special action required to start the DMA channel.

## DMA Channel Descriptor On Demand List Mode

In this flow mode, multiple descriptors form a chained list in which each descriptor set contains a pointer to the next descriptor set, allowing greater flexibility in memory layout options. When the descriptor set is fetched, this pointer value is loaded into the DMA channels next descriptor pointer register.

At the end of the work unit, if the DMA channel has not detected an incoming trigger event, the current work unit is repeated. If the DMA channel receives an incoming trigger before completion of the work unit then a new descriptor set is fetched.

## Descriptor Sets

The [Descriptor List Mode Parameter and Descriptor Offsets](#) illustrates how a descriptor set must be structured in memory. Note that while the descriptor sets can be dispersed throughout memory and reside in different memory blocks, each descriptor of the descriptor set must reside in a contiguous section of memory in the format shown in the table. The values have the same order as the corresponding MMR offset addresses.

**Table 13-12:** Descriptor List Mode Parameter and Descriptor Offsets

**Table 13-13:**

| Descriptor Offset | Parameter Register |
|-------------------|--------------------|
| 0x00              | DMA_DSCPTR_NXT     |
| 0x04              | DMA_ADDRSTART      |
| 0x08              | DMA_CFG            |
| 0x0C              | DMA_XCNT           |
| 0x10              | DMA_XMOD           |
| 0x14              | DMA_YCNT           |
| 0x18              | DMA_YMOD           |

All other DMA channel registers not loaded as a result of the descriptor set fetch retain their previous values. All of the register's current values are reloaded between the descriptor set fetch and the start of the DMA operation for the work unit.

## Minimum Startup Requirements

At a minimum the DMA\_DSCPTR\_NXT register must be written prior to write to the DMA\_CFG register which is the special action required to start the DMA channel.

## DMA Channel Data Transfer Modes

The DMA channel, in addition to supporting basic one-dimensional DMA transfers, also supports two-dimensional functionality.

## DMA Channel Two-Dimensional DMA

Register-based and descriptor-based DMA flow modes support two-dimensional data transfers.

In two-dimensional (2D) mode the X directional count and modifier (DMA\_XCNT and DMA\_XMOD) registers are accompanied by the Y directional count and modifier (DMA\_YCNT and DMA\_YMOD) registers, supporting arbitrary row and column sizes. Furthermore, modify values can be negative, allowing implementation of interleaved data streams. DMA\_XCNT and DMA\_YCNT specify the row and column sizes respectively, where the DMA\_XCNT must be 2 or greater.

The DMA start address (DMA\_STARTADDR) register, along with DMA\_XMOD and DMA\_YMOD registers, are all specified in bytes, and they must be aligned to a multiple of the DMA transfer word size as configured by the DMA\_CFG.MSIZE bit. Misalignment results in a DMA channel error.

The DMA\_XMOD register value is the byte-address increment that is applied after each transfer that decrements the DMA\_XCNT register. The DMA\_XCNT register is not applied when the inner loop count is ended by the DMA\_XCNT\_CUR register decrementing to 0 from 1, except that it is applied on the final transfer when the DMA\_YCNT register is 1 and the DMA\_XCNT register decrements from 1 to 0.

The DMA\_YMOD register value is the byte-address increment that is applied after each decrement of the value in the DMA\_YCNT\_CUR register. However, the DMA\_YMOD value is not applied to the last item in the array on which the outer loop count (DMA\_YCNT\_CUR) also expires by decrementing from 1 to 0.

After the last transfer completes, DMA\_YCNT\_CUR is 1 and the DMA\_XCNT\_CUR register is 0. The DMA channels current address points to the last items address plus the DMA\_XMOD register value. Note that if the DMA channel is programmed to refresh automatically such as in autobuffer mode, then both the DMA\_XCNT\_CUR and DMA\_YCNT\_CUR registers and the DMA current address (DMA\_ADDR\_CUR) are reloaded for the first data transfer of the next work unit.

Interrupt notification is configurable for end of row or end of work unit completion.

## DMA Channel Event Control

The DMA channel supports a number of events to provide notification of work unit state, peripheral data request, peripheral data request and completion events, and DMA channel error conditions. In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

The DMA channel has two interrupt signals for support of a number of events such as work unit state events, Peripheral Interrupt Request (PIRQ) events, Peripheral Data Request (PDR) events and DMA channel errors. DMA channel errors are reported on a dedicated interrupt signal while all other interrupt sources share the same interrupt signal. In addition to flexible interrupt configuration, the DMA channel also supports incoming and outgoing triggers which are useful in synchronizing the DMA channel with other system resources.

DMA channel events can be signaled to the processor using status information and optional interrupt requests. These events may be used for progress updates of data transfers and to request intervention from the processor core. A majority of DMA channel interrupts are configured by the respective bits in the DMA\_CFG register. Dedicated bits in the DMA\_STAT register are used to report the occurrence of various events. Interrupt requests are cleared by write-one-to-clear (W1C) operations to the status register.

**NOTE:** Hardware does not clear the interrupt status bits automatically even if the DMA channel is disabled then re-enabled. In this situation the interrupt signal from the DMA channel is de-asserted once the DMA channel is disabled, but the status bit remains set until the DMA channel is enabled again or cleared by software.

The DMA channel supports the following categories of events on the interrupt signals.

- Work unit state events are used to generate interrupts on row or on work unit DMA completion.
- Peripheral Interrupt Request (PIRQ) events are signaled by the peripheral when it has completed the transfer of all data.
- Peripheral Data Request (PDR) events for when the DMA channel is disabled or idle and the peripheral is requesting data from the DMA channel.
- Error events due to a failure in the work unit.

**ATTENTION:** The DMA channel will not generate an interrupt to the processor for a work unit state event, PIRQ event or forward a PDR event while in an error state.

## Event Signals

The *Event Signals* table provides descriptions of DMA channel events

**Table 13-14:** Event Signals

| Signal        | Width (bits) | Description                                                                                                                                                                                                       |
|---------------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DMA_ERROR     | 1            | Used to signal an error condition in the DMA channel. The source of the error can be determined by reading DMA_STAT.ERRC                                                                                          |
| DONE_PIRQ_INT | 1            | Signal used to indicate DMA completions events, PIRQ events and also for forwarding PDR events based on configuration. The source of the event may be determined by reading the corresponding fields in DMA_STAT. |
| DMA_TRIG_OUT  | 1            | Trigger output that gets routed to the TRU and can be configured to provide notification on row or work unit completion.                                                                                          |
| DMA_TRIG_IN   | 1            | Trigger input from the TRU that can be used to control the start of a work unit.                                                                                                                                  |

## Work Unit State Events

Work unit state events are generated as a result of a row or a work unit completion. In order for either of these events to result in the generation of an interrupt, the interrupt of the DMA channel must be configured for one of the available work unit completion modes.

- Current X count reaching 0 for row completion or 1-D DMA work unit completion.
- Current Y count reaching 0 for work unit completion of 2-D DMA.

**NOTE:** For one -dimensional DMA, configuring the interrupt to be generated on the current Y counter reaching 0 results in a DMA channel configuration error.

The DMA channel issues the last memory read or write transaction for the row or work unit and then pauses until the read or write acknowledge is returned. Once the transfer has been acknowledged successfully the DMA channel issues the interrupt and continues to process the next row or work unit.

Waiting for the memory access to be acknowledged results in a delay. However, programs can read or modify data in the memory without adversely affecting, or being affected by, the DMA transfer.

**NOTE:** While the DMA channel may be paused waiting for the memory transfer to be acknowledged, the DMA channel is still capable of fetching the next descriptor set in order to be ready to process the next work unit as soon as the memory access completes.

The interrupt timing is also affected by the synchronization feature of the DMA channel's configuration. For memory read operations with synchronization enabled, the interrupt is delayed further until the last transfer from the DMA channel's FIFO to the peripheral completes. The interrupt timing for memory write operations is not affected by the synchronization feature.

## Peripheral Interrupt Request Events

Peripheral Interrupt Request (PIRQ) events may be used by a peripheral connected to the DMA channel to indicate, in the case of a peripheral transmit operation, that data has not only left the FIFO of the DMA channel, but that the peripheral has also completed the transfer.

In order to support PIRQ interrupts the DMA channel's interrupt must be configured accordingly. This disables the generation of interrupts based on work unit state and instead results in the generation an interrupt when the DMA channel receives the command from the peripheral.

The interrupt is only generated if the following conditions are satisfied.

- The DMA channel is enabled
- The DMA channel is in the stop state
- The DMA channel's interrupt is configured for PIRQ operation

## Peripheral Data Request Events

Peripheral Data Request (PDR) events occur when an interfaced peripheral requests data from the DMA channel and the DMA channel is either disabled or enabled and in the stop state.

When the DMA channel is disabled and a peripheral sends a command to the DMA channel to request data, the DMA channel generates an interrupt to the System Event Controller (SEC). There is no status information reported about this event in the DMA channel's status register. Instead, the PDR event is identified by the fact that the DMA channel generated an interrupt when it was disabled. Further confirmation can be obtained by verifying the status of the peripheral interfaced to the DMA channel.

In addition to requests for data being forwarded as interrupts when the DMA channel is in the disabled state, the DMA channel is also able to forward PDR events as an interrupt when the DMA channel is in the stop state after completion of a work unit. The forwarding of this interrupt when the DMA channel is in the stop state is optional and configured by the program during DMA channel configuration.

## DMA Channel Triggers

DMA channel triggers are useful for synchronizing the DMA channel with other events in the system. Channel triggers can be used in combination with each other in order to create ping-pong buffers or when combined with interrupts to notify the processor that a particular milestone has been reached and that service is required. Triggers may also be used to enforce a handshake DMA operation in which the trigger acts as a signal for a DMA request.

**NOTE:** Using the trigger to control the pace of data transfers, such as in the case of a handshake DMA, requires that all the data for the entire work unit is ready for transfer.

The DMA channel has a single incoming trigger that can be used to control the pace of the data transfers performed by the DMA channel. The DMA channel can be configured to wait for the incoming trigger before starting the work unit transfer or fetching a descriptor set from memory.

The DMA channel also has a single outgoing trigger signal that may be configured to signal the end of row or an entire work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, and then pauses until the transfer acknowledge is returned. Once the transfer has been acknowledged, the DMA channel issues the trigger before processing the next row or work unit.

## Issuing Triggers

The DMA channel can be configured to generate an outgoing trigger signal at the end of row or the end of a work unit. The DMA channel issues the last memory read or memory write transaction for the row or work unit, and then pauses until the transfer acknowledge is returned. Once the transfer has been acknowledged the DMA channel issues the trigger before processing the next row or work unit.

**NOTE:** While the DMA channel may be paused while waiting for the memory transfer to be acknowledged, the DMA channel is still capable of fetching the next descriptor set in order to be ready to process the next work unit as soon as the memory access completes.



## Waiting For Triggers

Triggering may be used to control the pace of data transfers performed by the DMA channel. If `DMA_CFG.TWAIT` is enabled and a trigger has been received since the last time the DMA channel left the wait state or since transition from disabled to enabled, then the DMA channel will enter a wait state before beginning the next work unit. The DMA channel will not perform a descriptor fetch either. Once a trigger is received, the DMA channel will leave the wait state and begin the next work unit or fetch the next descriptor if configured for a descriptor based mode of operation.

If the DMA channel is programmed through a memory mapped register write operation with stop flow mode enabled, `DMA_CFG.TWAIT` enabled and no trigger having already been received, then the DMA channel will enter a wait state before performing the data transfer. Upon receiving the trigger, the DMA channel will begin the data transfer portion of the work unit. Once the data transfer is complete, the DMA channel will enter the stop state.

If the DMA channel is programmed through a memory mapped register write operation with the flow mode configured to one of the descriptor based modes, then the DMA channel will enter wait state before performing the descriptor fetch. Once the descriptor fetch is complete, the DMA channel immediately proceeds to the data transfer, regardless of the value of the `DMA_CFG.TWAIT` bit. If the descriptor fetch is followed by another descriptor fetch, then the DMA channel enters a wait state before fetching the next descriptor.

If the descriptor fetch returns a descriptor with stop flow mode then the `DMA_CFG.TWAIT` value for that descriptor will not affect the DMA as the DMA channel will enter the stop state once the data transfer is completed. The DMA channel only enters the wait state based on `DMA_CFG.TWAIT` before the next work unit or descriptor fetch.

If the descriptor fetch returns a descriptor configured for Autobuffer flow mode, then `DMA_CFG.TWAIT` for that descriptor will not affect the DMA for the first work unit of the autobuffer transfer. Once the first work unit is completed another trigger has not been received, then the DMA channel will enter the wait state before re-initializing its counters and address registers if not configured for current addressing. The next work unit is performed once the trigger is received.

The incoming trigger does not have to be issued after the DMA channel has entered the wait state, and can be issued whilst the DMA channel is in the midst of executing a work unit, descriptor fetch or even when in the stop state. the trigger will be held internally, and once the work unit is complete, the DMA channel will skip the wait state and proceed directly to executing the following work unit. If `DMA_CFG.TWAIT` is not enabled, the DMA channel will also skip the wait state, however the trigger will be held internally and used the next time `DMA_CFG.TWAIT` is enabled. This allows the user to enable the `DMA_CFG.TWAIT` functionality several work units apart and not be concerned with losing a trigger. The DMA channels trigger overrun enable functionality may be enabled in all work units to ensure multiple triggers do not occur between the work units with `DMA_CFG.TWAIT` enabled.

## DMA Channel Programming Model

Several synchronization and control methods are available for use in development of software tasks which manage peripheral DMA and memory DMA. Such software needs to be able to accept requests for new DMA transfers from other software tasks, integrate these transfers into existing transfer queues, and reliably notify other tasks when the transfers are complete.

In the processor, it is possible for each peripheral DMA and memory DMA stream to be managed by a separate task or to be managed together with any other stream. Each DMA channel has independent, orthogonal control registers, resources, and interrupts, so that the selection of the control scheme for one channel does not affect the choice of control scheme on other channels. For example, one peripheral can use a linked-descriptor-list, interrupt-driven scheme while another peripheral can simultaneously use a demand-driven, buffer-at-a-time scheme synchronized by polling of DMA events.

The topics that follow describe the steps required to configure the DMA channel for the various modes in addition to the programming concepts required for software synchronization.

## DMA Channel Mode Configuration

Use the step-by-step directions that follow to set up the DMA channel for operating modes. These directions include flowcharts and links to reference information where appropriate.

### Register Based Linear Buffer Stop Flow Mode

Configures a peripheral's DMA channel to read data from internal memory and send it to the peripheral for transmission.

#### *PREREQUISITE:*

The peripheral is assumed to be in a state where it is ready to transmit data received from the DMA channel.

The task involves writing to a number of DMA channel MMR registers in order to configure a DMA channel to read data from internal memory and send it to a peripheral connected to the peripheral DMA bus. Upon DMA completion the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the `DMA_ADDRSTART` register.

*ADDITIONAL INFORMATION:* The address may be used to calculate the most optimum `DMA_CFG.MSIZE` possible.

2. Calculate most optimum `DMA_CFG.MSIZE` based on the `DMA_ADDRSTART` register and number of bytes in work unit.

*ADDITIONAL INFORMATION:* The number of bytes in the work unit must be a multiple of the selected `DMA_CFG.MSIZE` and the start address alignment must also be considered.

3. Write the `DMA_XCNT` register based on calculated `DMA_CFG.MSIZE`.

*ADDITIONAL INFORMATION:* `DMA_XCNT` is the number of `DMA_CFG.MSIZE` transfers required to make up the entire work unit.

4. Write the `DMA_XMOD` register.

*ADDITIONAL INFORMATION:* For a completely linear buffer transfer, `DMA_XMOD` is determined by the selected `DMA_CFG.MSIZE`. This register is always specified in the number of bytes.

5. Write the `DMA_CFG` register with `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for STOP mode, `DMA_CFG.WNR` must be configured for memory read operation and `DMA_CFG.PSIZE` must be configured to a value no larger than the supported bus width of the peripheral DMA bus.

- `DMA_CFG.SYNC` may be configured to control DMA completion notification timing.
- Interrupts and triggers may also be configured at this step depending on requirements.

*RESULT:*

The DMA channel is now enabled and the buffer is transferred. The DMA channel enters the IDLE state upon completion of the work unit.

## Register Based Autobuffer Flow Mode

Configures a peripherals' DMA channel to read data from internal memory and send it to the peripheral for transmission. The transmission of the buffer is repeated endlessly.

*PREREQUISITE:*

The peripheral is assumed to be in a state where it is ready to transmit data received from the DMA channel

The task involves writing to a number of DMA channel MMR registers in order to configure a DMA channel to read data from internal memory and send it to a peripheral connected to the peripheral DMA bus. Upon DMA completion the DMA channel starts the DMA operation over again creating an endless circular buffer transfer.

1. Write the `DMA_ADDRSTART` register.

*ADDITIONAL INFORMATION:* The address may be used to calculate the most optimum `DMA_CFG.MSIZE` possible.

2. Calculate most optimum `DMA_CFG.MSIZE` based on the `DMA_ADDRSTART` register and number of bytes in work unit.

*ADDITIONAL INFORMATION:* The number of bytes in the work unit must be a multiple of the selected `DMA_CFG.MSIZE` and the start address alignment must also be considered.

3. Write the `DMA_XCNT` register based on calculated `DMA_CFG.MSIZE`.

*ADDITIONAL INFORMATION:* `DMA_XCNT` is the number of `DMA_CFG.MSIZE` transfers required to make up the entire work unit.

4. Write the `DMA_XMOD` register.

*ADDITIONAL INFORMATION:* For a completely linear buffer transfer, `DMA_XMOD` is determined by the selected `DMA_CFG.MSIZE`. This register is always specified in the number of bytes.

5. Write the `DMA_CFG` register with the `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for AUTOBUFFER mode, `DMA_CFG.WNR` must be configured for memory read operation and `DMA_CFG.PSIZE` must be configured to a value no larger than the supported bus width of the peripheral DMA bus.

- `DMA_CFG.SYNC` may be configured to control DMA completion notification timing.
- Interrupts and triggers may also be configured at this step depending on requirements.

*RESULT:*

The DMA channel is now enabled and the buffer is transferred until the DMA channel is disabled.

## Descriptor Array Flow Mode

Configures a peripheral's DMA channel to read data from memory as described by the descriptor sets in the array and send it to the peripheral for transmission. Descriptor sets are read from an array in memory to configure the individual work units.

*PREREQUISITE:*

The peripheral is assumed to be in a state where it is ready to transmit data received from the DMA channel. The array of descriptors is assumed to be initialized with the last descriptor set configured for STOP flow mode.

The task involves writing to a number of DMA channel MMR registers in order to configure a DMA channel to read the first descriptor set from the array in memory that is responsible for the configuring the DMA channel to retrieve and send the data to a peripheral connected to the peripheral DMA bus. Upon

DMA completion the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the `DMA_DSCPTR_CUR` register with the address of the array in which the descriptor sets are stored.

*ADDITIONAL INFORMATION:* The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the `DMA_CFG` register with the `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for DESCRIPTOR ARRAY mode, `DMA_CFG.NDSIZE` must be configured to describe the number of descriptor elements contained within the first descriptor set. `DMA_CFG.WNR` must be configured for memory read operation and `DMA_CFG.PSIZE` must be configured to a value no larger than the supported bus width of the peripheral DMA bus.

- `DMA_CFG.SYNC` configuration is controlled by the descriptor set that is to be fetched as are interrupt and trigger configurations

*STEP RESULT:* The first descriptor set is fetched from memory location provided by `DMA_DSCPTR_CUR` and loaded to the DMA Channels MMR registers.

*RESULT:*

The DMA channel is now processing all the work units provided in the descriptor array. The DMA channel enters the IDLE state upon completion of the final work unit that was configured for STOP flow mode.

## Descriptor List Flow Mode

Configures a peripheral's DMA channel to read data from memory as described by the descriptor sets in the list and send it to the peripheral for transmission. Descriptor sets are read from a list of descriptors in which each descriptor set has a descriptor that points to the next descriptor set location in memory.

*PREREQUISITE:*

The peripheral is assumed to be in a state where it is ready to transmit data received from the DMA channel. The list of descriptors is assumed to be initialized with the last descriptor set in the list configured for STOP flow mode.

The task involves writing to a number of DMA channel MMR registers in order to configure a DMA channel to read the first descriptor set from the list in memory that is responsible for the configuring the DMA channel to retrieve and send the data to a peripheral connected to the peripheral DMA bus. Upon

DMA completion the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the `DMA_DSCPTR_NXT` register with the address of the first descriptor in the list to be processed.

*ADDITIONAL INFORMATION:* The array address must meet any processor alignments restrictions imposed by descriptor fetches.

2. Write the `DMA_CFG` register with the `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for DESCRIPTOR LIST mode, and the `DMA_CFG.NDSIZE` bit must be configured to describe the number of descriptor elements contained within the first descriptor set. The `DMA_CFG.WNR` bit must be configured for memory read operation and the `DMA_CFG.PSIZE` bit must be configured to a value no larger than the supported bus width of the peripheral DMA bus.

- `DMA_CFG.SYNC` configuration is controlled by the descriptor set that is to be fetched as are interrupt and trigger configurations.

*STEP RESULT:* The first descriptor set is fetched from the memory location provided by `DMA_DSCPTR_NXT` and loaded to the DMA channel's MMR registers.

*RESULT:*

The DMA channel is now processing all the work units provided in the descriptor list. The DMA channel enters the IDLE state upon completion of the final work unit that was configured for STOP flow mode.

## Register Based Memory to Memory Transfer in Stop Flow Mode

Configures a memory DMA channel pair in STOP flow mode. One DMA channel is configured for memory read operations while the other is configured for memory write.

*PREREQUISITE:*

The task involves writing to a number of DMA channels on two DMA channels that create a memory DMA pair. Upon DMA completion the DMA channel enters the idle state until either disabled or reconfigured for a new transfer.

1. Write the `DMA_ADDRSTART` register of the source DMA channel.

*ADDITIONAL INFORMATION:* The address may be used to calculate the most optimum `DMA_CFG.MSIZE` possible.

2. Calculate most optimum `DMA_CFG.MSIZE` based on the `DMA_ADDRSTART` register and number of bytes in work unit.

*ADDITIONAL INFORMATION:* The number of bytes in the work unit must be a multiple of the selected `DMA_CFG.MSIZE` and the start address alignment must also be considered.

3. Write the `DMA_XCNT` register of the source DMA channel based on calculated `DMA_CFG.MSIZE`.

*ADDITIONAL INFORMATION:* `DMA_XCNT` is the number of `DMA_CFG.MSIZE` transfers required to make up the entire work unit.

4. Write the `DMA_XMOD` register of the source DMA channel.

*ADDITIONAL INFORMATION:* For a completely linear buffer transfer, `DMA_XMOD` is determined by the selected `DMA_CFG.MSIZE`. This register is always specified in the number of bytes.

5. Write the `DMA_ADDRSTART` register of the destination DMA channel.

*ADDITIONAL INFORMATION:* The address may be used to calculate the most optimum `DMA_CFG.MSIZE` possible.

6. Calculate most optimum `DMA_CFG.MSIZE` based on the `DMA_ADDRSTART` register and number of bytes in work unit.

*ADDITIONAL INFORMATION:* The number of bytes in the work unit must be a multiple of the selected `DMA_CFG.MSIZE` and the start address alignment must also be considered.

7. Write the `DMA_XCNT` register of the destination DMA channel based on calculated `DMA_CFG.MSIZE`.

*ADDITIONAL INFORMATION:* `DMA_XCNT` is the number of `DMA_CFG.MSIZE` transfers required to make up the entire work unit.

8. Write the `DMA_XMOD` register of the destination DMA channel.

*ADDITIONAL INFORMATION:* For a completely linear buffer transfer, `DMA_XMOD` is determined by the selected `DMA_CFG.MSIZE`. This register is always specified in the number of bytes.

9. Write the `DMA_CFG` register of the source DMA channel with `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for STOP mode, `DMA_CFG.WNR` must be configured for memory read operation and `DMA_CFG.PSIZE` must be configured to a value no larger than the supported bus width of the peripheral DMA bus.

- `DMA_CFG.SYNC` may be configured to control DMA completion notification timing.
- Interrupts and triggers may also be configured at this step depending on requirements, generally however they would be enabled within the destination DMA channel configuration.

*STEP RESULT:* The memory read DMA transfer begins.

10. Write the `DMA_CFG` register of the destination DMA channel with `DMA_CFG.EN` configured to enable the DMA channel.

*ADDITIONAL INFORMATION:* `DMA_CFG.FLOW` must be set for STOP mode, `DMA_CFG.WNR` must be configured for memory write operation and `DMA_CFG.PSIZE` must be configured to a value no larger than the

supported bus width of the peripheral DMA bus. This must also match the value written for the source DMA channel configuration.

- Interrupts and triggers may also be configured at this step depending on requirements.

*STEP RESULT:* The memory write DMA transfer begins.

*RESULT:*

Both memory DMA channels are now running and the data is transferred from the source address to the destination address. The DMA channel enters the IDLE state upon completion of the work unit.

## **DMA Channel Programming Concepts**

Using the features, operating modes, and event control for the DMA channel to their greatest potential requires an understanding of some DMA channel related concepts.

### **Synchronization of Software and DMA**

A critical element of software DMA management is synchronization of DMA work unit completion with the software. This can best be achieved using DMA channel interrupt and trigger events, or through polling of these events status bits within the DMA channel registers, or a combination of both. Polling for address or count can only provide synchronization within loose tolerances comparable to pipeline lengths.

#### **Interrupt and Trigger Event Based Synchronization**

Interrupt and trigger based synchronization methods must avoid interrupt/trigger overrun, or the failure to invoke a DMA channel's event handler for every event due to excessive latency in processing of events. Generally, the system design must either ensure that only one event per channel is scheduled (for example, at the end of a descriptor list), or that generated events are spaced sufficiently far apart in time that system processing budgets can guarantee every event is serviced.

When the DMA channel issues an interrupt/trigger event or changes event status bits in the DMA\_STAT register, it guarantees that the last memory operation of the work unit is complete. For memory read DMA transactions, this means that the final memory read data has been safely received in the DMA channel's FIFO. For memory write DMA transactions, this means that the DMA channel has received an acknowledge that the last write transfer of the work unit has completed.

#### **Register Polling Based Synchronization**

Polling of DMA channel registers such as the DMA\_ADDR\_CURR, DMA\_DSCPTR\_CURR, or DMA\_XCNT\_CURR/DMA\_YCNT\_CURR registers is not recommended as a method of precisely synchronizing DMA with data processing, due to DMA channel FIFOs and DMA/memory pipelining. The current address, pointer, and count registers change several cycles in advance of the completion of the corresponding memory operation, as measured by the time at which the results of the operation are first visible to the core by memory read or write instructions.



For example, in a DMA channel memory write operation to external memory, assume a DMA channel write operation is initiated by DMA channel A. This causes the DDR SDRAM to perform a page open operation which takes many system clock cycles. Meanwhile, another DMA channel (channel B) which does not in itself incur latency, initiates a transfer that is stalled behind the slow operation of channel A. Software monitoring channel B could not safely conclude whether the memory location pointed to by channel B's `DMA_ADDR_CURR` register has or has not been written, based solely on this register contents.

Polling of the current address, pointer, and count registers can permit loose synchronization of DMA with software if allowances are made for the lengths of the DMA/memory pipeline. Further, the length of the DMA FIFO for a particular peripheral needs to be taken into consideration. The DMA channel does not advance current address/pointer/count registers if these FIFOs are filled with incomplete work (including reads that have been started but not yet finished).

Additionally, the length of the pipelines to the destination memory needs to be taken into consideration. If the DMA FIFO length and the DMA channel's memory pipeline length are added, an estimate can be made of the maximum number of incomplete memory operations in progress at one time.

*The estimate would be a maximum, as the DMA/memory pipeline may include traffic from other DMA channels.*

## Descriptor Queues

A system designer might want to write a DMA manager facility which accepts DMA requests from other software. The DMA manager software does not know in advance when new work requests are received or what these requests might contain. The software could manage these transfers using a circular linked list of DMA descriptors, where each descriptor sets `DMA_DSCPTR_NXT` descriptor points to the next descriptor set, and the last descriptor set points to the first.

The code that writes into this descriptor list could use the processor's circular addressing modes, so that it does not need to use comparison and conditional instructions to manage the circular structure. In this case, the `DMA_DSCPTR_NXT` descriptor of each descriptor set could even be written once at startup, and skipped over as each descriptor's new contents are written.

The recommended method for synchronization of a descriptor queue is through the use of an interrupt or trigger. The descriptor queue is structured so that at least the final valid descriptor set is always programmed to generate an interrupt or trigger event upon completion.

- [Queues Using Event Generation for Every Descriptor Set](#)
- [Queues Using Minimal Events](#)

### Queues Using Event Generation for Every Descriptor Set

In this system, the DMA manager software synchronizes with the DMA channel by enabling an interrupt or trigger on every descriptor set. This method should only be used if the system design can guarantee that each work unit completion event is serviced separately (no interrupt or trigger overrun).

To maintain synchronization of the descriptor set queue, the non-interrupt software maintains a count of descriptor sets added to the queue, while the event handler (either interrupt or trigger) maintains a count of completed descriptor sets removed from the queue. The counts are equal only when the DMA channel is paused after having processed all the descriptor sets.

When each new work unit event is received, the DMA manager software initializes a new descriptor set, taking care to set the flow to STOP mode. Next, the software compares the descriptor set counts to determine if the DMA channel is running or not. If the DMA channel is paused (counts equal), the software increments its count and then starts the DMA channel by writing the new descriptor sets' DMA\_CFG descriptor.

If the counts are unequal, the software instead modifies the next-to-last descriptor sets' DMA\_CFG descriptor so that it now describes the newly-queued descriptor set. This operation does not disrupt the DMA channel provided the rest of the descriptor sets' descriptors are initialized in advance. It is necessary, however, to synchronize the software to the DMA to correctly determine whether the new or the old DMA\_CFG value was read by the DMA channel.

The synchronization operation should be performed in the event handler. First, when an event is detected, the handler should read the channel's DMA\_STAT register. If DMA\_STAT.RUN indicates the DMA channel is running, then the channel has moved on to processing another descriptor, and the event handler may increment its count and exit. If DMA\_STAT.RUN indicates the channel is not running, then the channel is paused, either because there are no more descriptor sets to process, or because the last descriptor set was queued too late (that is, the modification of the next-to-last descriptor sets' DMA\_CFG descriptor occurred after that descriptor was read into the DMA channel). In this case, the event handler writes the DMA\_CFG value appropriate for the last descriptor set to the DMA channel's DMA\_CFG register, increments the completed descriptor count, and exits.

Again, this system can fail if the system's event latencies are large enough to cause any of the channel's DMA events to be dropped. An event handler capable of safely synchronizing multiple descriptor set interrupts is complex, performing several MMR accesses to ensure robust operation. In such a system environment a minimal event synchronization method is preferred.

## **Queues Using Minimal Events**

In this system, only one DMA interrupt or trigger event is generated in the queue at any time. The DMA event handler for this system can also be extremely short. Here, the descriptor queue is organized into an *active* and a *waiting* portion, where events are enabled only on the last descriptor set in each portion.

When each new DMA request is processed, the software fills in a new descriptor set's contents and adds it to the waiting portion of the queue. The descriptor set's DMA\_CFG descriptor should have the flow set to Stop mode. If more than one request is received before the DMA queue completion event occurs, the non-interrupt code queues later descriptor sets, forming a waiting portion of the queue that is disconnected from the active portion of the queue being processed by the DMA channel. In other words, all but the last active descriptor sets contain FLOW values for a descriptor based mode and have no event enable set. The last active descriptor set has the Stop flow mode and an event generation enabled. Also, all but the last waiting descriptor sets are configured for descriptor based flow modes with no event generation. Only the last waiting descriptor set is configured for Stop flow mode and event generation enabled. This ensures

that the DMA channel can automatically process the whole active queue before then issuing one event. Also, this arrangement makes it easy to start the waiting queue within the event handler by a single `DMA_CFG` register write.

After queuing a new waiting descriptor, the non-interrupt software leaves a message for its interrupt handler in a memory mailbox location containing the desired `DMA_CFG` value to use to start the first waiting descriptor set in the waiting queue (or 0 to indicate no descriptors are waiting).

It is critical that the software not modify the contents of the active descriptor set queue directly, once processing by the DMA channel has started, unless careful synchronization measures are taken. In the most straightforward implementation of a descriptor set queue, the DMA manager software never modifies descriptors on the active queue; instead, the DMA manager waits until the DMA queue completion event indicates the processing of the entire active queue is complete.

When a DMA queue completion event is received, the event handler reads the mailbox from the non-interrupt software and writes the value to the DMA channel's `DMA_CFG` register. This single register write restarts the queue, effectively transforming the waiting queue to an active queue. The event handler then passes a message back to the non-interrupt software indicating the location of the last descriptor set accepted into the active queue. If, on the other hand, the event handler reads its mailbox and finds a `DMA_CFG` value of zero, indicating there is no more work to perform, then it passes an appropriate message back to the non-interrupt software indicating that the queue has stopped.

The non-interrupt software which accepts new DMA work unit requests needs to synchronize the activation of a new work unit with the interrupt handler. If the queue has stopped (that is, if the mailbox from the event handler is zero), the non-interrupt software is responsible for starting the queue (writing the first descriptor sets `DMA_CFG` value to the channel's `DMA_CFG` register). If the queue is not stopped, the non-interrupt software must not write the `DMA_CFG` register (which would cause a DMA error), but instead it should queue the descriptor onto the waiting queue and update its mailbox directed to the event handler.

## DMA Register Descriptions

DMA Channel (DMA) contains the following registers.

**Table 13-15:** DMA Module Registers

| Name                        | Description                        |
|-----------------------------|------------------------------------|
| <code>DMA_DSCPTR_NXT</code> | Pointer to Next Initial Descriptor |
| <code>DMA_ADDRSTART</code>  | Start Address of Current Buffer    |
| <code>DMA_CFG</code>        | Configuration Register             |
| <code>DMA_XCNT</code>       | Inner Loop Count Start Value       |

**Table 13-15:** DMA Module Registers (Continued)

| Name           | Description                              |
|----------------|------------------------------------------|
| DMA_XMOD       | Inner Loop Address Increment             |
| DMA_YCNT       | Outer Loop Count Start Value (2D only)   |
| DMA_YMOD       | Outer Loop Address Increment (2D only)   |
| DMA_DSCPTR_CUR | Current Descriptor Pointer               |
| DMA_DSCPTR_PRV | Previous Initial Descriptor Pointer      |
| DMA_ADDR_CUR   | Current Address                          |
| DMA_STAT       | Status Register                          |
| DMA_XCNT_CUR   | Current Count(1D) or intra-row XCNT (2D) |
| DMA_YCNT_CUR   | Current Row Count (2D only)              |
| DMA_BWLCNT     | Bandwidth Limit Count                    |
| DMA_BWLCNT_CUR | Bandwidth Limit Count Current            |
| DMA_BWMCNT     | Bandwidth Monitor Count                  |
| DMA_BWMCNT_CUR | Bandwidth Monitor Count Current          |

## Pointer to Next Initial Descriptor

The DMA\_DSCPTR\_NXT register specifies the start location of the next descriptor set, which begins when the DMA activity specified by the current descriptor set completes. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

The DMA\_DSCPTR\_NXT register is only used in descriptor list mode. At the start of a descriptor fetch in this mode, the DMA\_DSCPTR\_NXT register is copied into the DMA\_DSCPTR\_CUR register. During descriptor fetch, the DMA increments the DMA\_DSCPTR\_CUR register value after reading each element of the descriptor set.

In descriptor list mode, the DMA\_DSCPTR\_NXT register (not the DMA\_DSCPTR\_CUR register) must be programmed directly through MMR access, before the DMA operation is started. In descriptor array mode, the DMA disregards the DMA\_DSCPTR\_NXT register and uses the DMA\_DSCPTR\_CUR register to control descriptor fetch.

DMA\_DSCPTR\_NXT: Pointer to Next Initial Descriptor - R/W

Reset = 0x0000 0000

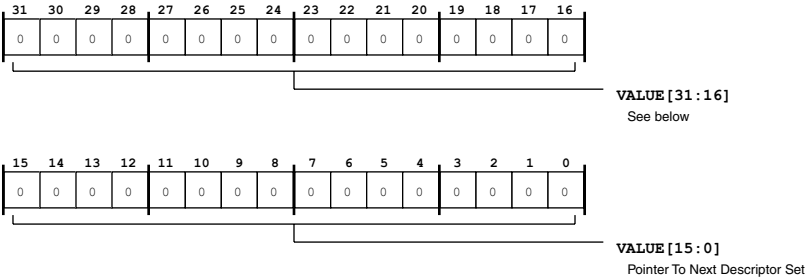


Figure 13-4: DMA\_DSCPTR\_NXT Register Diagram

Table 13-16: DMA\_DSCPTR\_NXT Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 31:0    | VALUE    | Pointer To Next Descriptor Set. |

Start Address of Current Buffer

The DMA\_ADDRSTART register contains the start address of the Work Unit currently targeted for DMA. This register is read/write prior to enabling the channel, but is read-only after enabling channel.

DMA\_ADDRSTART: Start Address of Current Buffer - R/W

Reset = 0x0000 0000

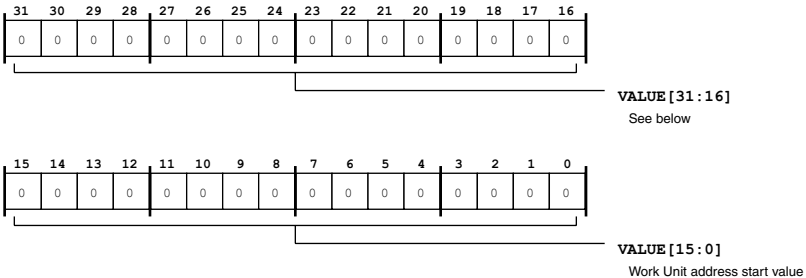


Figure 13-5: DMA\_ADDRSTART Register Diagram

Table 13-17: DMA\_ADDRSTART Register Fields

| Bit No. | Bit Name | Description/Enumeration        |
|---------|----------|--------------------------------|
| 31:0    | VALUE    | Work Unit address start value. |

Configuration Register

The DMA\_CFG sets up DMA parameters and operation modes. Other than clearing the DMA\_CFG.EN bit, writing to the DMA\_CFG register while a DMA process is already running cause a DMA error.

DMA\_CFG: Configuration Register - R/W

Reset = 0x0000 0000

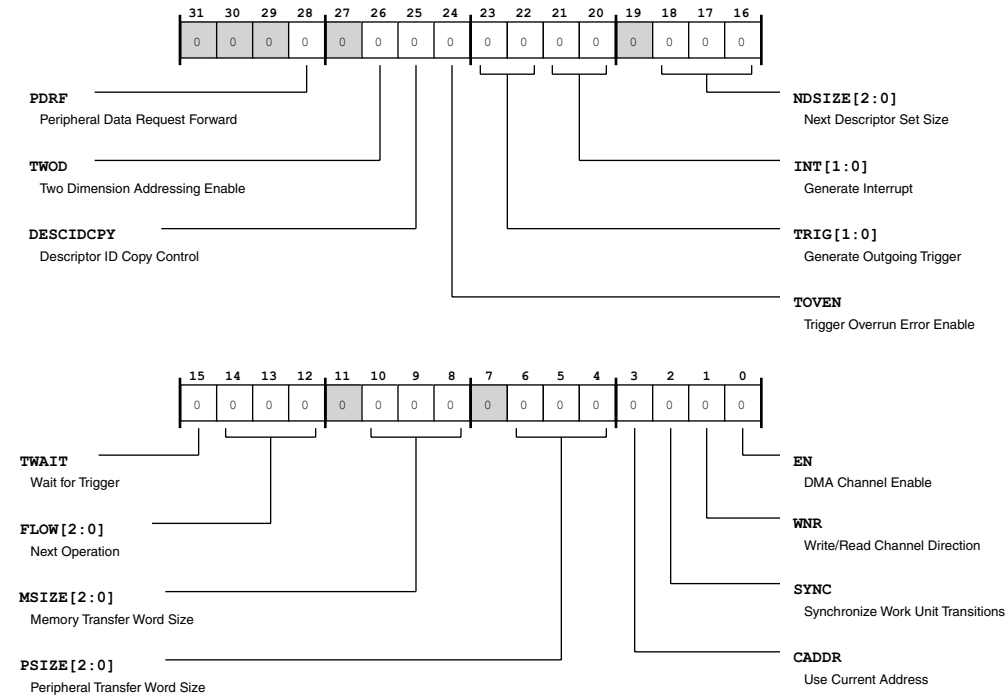


Figure 13-6: DMA\_CFG Register Diagram

Table 13-18: DMA\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28      | PDRF     | Peripheral Data Request Forward.<br>The DMA_CFG.PDRF defines how the DMA handles data requests from the peripheral while in idle state after a stop mode or memory read work unit. If set, the DMA forwards the peripheral data request as an interrupt.<br>Note that the peripheral data request forward selection applies only to DMA_CFG.FLOW bits set for stop and DMA_CFG.WNR bits set for memory read. |
|         |          | 0 Peripheral Data Request Not Forwarded                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Peripheral Data Request Forwarded                                                                                                                                                                                                                                                                                                                                                                          |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | TWOD      | Two Dimension Addressing Enable.<br>The DMA_CFG.TWOD selects whether the DMA addressing involves only DMA_XCNT and DMA_XMOD (one-dimensional DMA) or also involves DMA_YCNT and DMA_YMOD (two-dimensional DMA).                                                                                                                                                                                                                                   |
|         |           | 0 One-Dimensional Addressing                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 1 Two-Dimensional Addressing                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 25      | DESCIDCPY | Descriptor ID Copy Control.<br>The DMA_CFG.DESCIDCPY specifies when to copy the initial descriptor pointer to the DMA_DSCPTR_PRV register.<br>Note that a bus write to DMA_CFG to clear the DMA_CFG.EN bit cause the DMA to use the new value of DMA_CFG.DESCIDCPY immediately. To preserve consistency (if required by application), the new value of DMA_CFG.DESCIDCPY should match the previous value.                                         |
|         |           | 0 Never Copy                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 1 Copy on Work Unit Complete                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 24      | TOVEN     | Trigger Overrun Error Enable.<br>A trigger overrun occurs if more than one trigger was received before the DMA reached the trigger wait state. If DMA_CFG.TOVEN is set, a trigger overrun causes the DMA to flag an error. In cases where a trigger overrun is not a problem, clearing DMA_CFG.TOVEN prevents the overrun from causing an error and halting the DMA. The DMA_CFG.TOVEN operates independently of the DMA_CFG.TWAIT bit selection. |
|         |           | 0 Ignore Trigger Overrun                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Error on Trigger Overrun                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:22   | TRIG     | <p>Generate Outgoing Trigger.</p> <p>The DMA_CFG.TRIG selects whether the DMA issues an outgoing trigger, based on the work unit counter values. In one-dimensional mode, the only options are to trigger at the end of the whole Work Unit (trigger when DMA_XCNT_CUR reaches 0) or not to trigger at all. If in one-dimensional addressing mode, programming DMA_CFG.TRIG to trigger when DMA_YCNT_CUR reaches 0 (or to reserved) cause the DMA to flag a configuration error.</p> <p>If in two-dimensional addressing mode, the options are to trigger at the end of each row of the inner loop (when DMA_XCNT_CUR reaches 0), to trigger only after completing the whole work unit (when DMA_YCNT_CUR reaches 0), or not to trigger at all. If in two-dimensional mode and set to trigger when DMA_XCNT_CUR reaches 0, the DMA also issues a trigger at the end of the work unit. If in two-dimensional addressing mode, programming DMA_CFG.TRIG to reserved causes the DMA to flag a configuration error.</p> <p>If DMA_CFG.TRIG is non-zero and the peripheral issues a finish command, the DMA issues a trigger after the finish procedure is complete.</p> <p>For more information about trigger generation timing, see the trigger section of the DMA functional description.</p> |
|         |          | 0 Never assert Trigger                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Trigger when XCNTCUR reaches 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 2 Trigger when YCNTCUR reaches 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 3 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |



**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| 21:20   | INT      | <p>Generate Interrupt.</p> <p>The <code>DMA_CFG.INT</code> selects whether an interrupt is sent to the core based on work unit status or peripheral interrupt request.</p> <p>For one-dimensional mode, setting <code>DMA_CFG.INT</code> for interrupt when <code>DMA_YCNT_CUR</code> reaches 0 causes the DMA to flag a configuration error.</p> <p>The peripheral interrupt setting enables the DMA to generate the last grant indication and to accept/forward the peripheral interrupt command.</p> <p>Note that the peripheral interrupt selection applies only to <code>DMA_CFG.FLOW</code> bits set for stop and <code>DMA_CFG.WNR</code> bits set for memory read.</p> <p>If <code>DMA_CFG.INT</code> is set for interrupt on count completion (<code>DMA_XCNT_CUR</code> or <code>DMA_YCNT_CUR</code> reach 0) and the peripheral issues a finish command, the DMA issues an interrupt after the finish procedure is complete.</p> <p>For more information see the sections on interrupt generation and peripheral control in the DMA functional description.</p> |                                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Never assert Interrupt         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Interrupt when X Count Expires |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Interrupt when Y Count Expires |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Peripheral Interrupt           |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | NDSIZE   | Next Descriptor Set Size.<br>The <code>DMA_CFG.NDSIZE</code> specifies the number of descriptor elements in memory to load during the next descriptor fetch. The DMA loads the descriptors in a specific order. The descriptor set may or may not have the next descriptor pointer, depending on whether it is a descriptor list or descriptor array.                                                                                                                                                                    |
|         |          | 0 Fetch one Descriptor Element                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Fetch two Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 Fetch three Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 3 Fetch four Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 4 Fetch five Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 5 Fetch six Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 6 Fetch seven Descriptor Elements                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 7 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 15      | TWAIT    | Wait for Trigger.<br>The <code>DMA_CFG.TWAIT</code> controls whether the DMA waits for a incoming trigger from another channel or user. If <code>DMA_CFG.TWAIT</code> is set, the DMA enters the wait state before starting the next work unit, including descriptor fetch if in descriptor mode. Using the wait for trigger control is not allowed for descriptor-on-demand mode, and using this control in that mode causes an error. For more information, see the trigger section of the DMA functional description. |
|         |          | 0 Begin Work Unit Automatically (No Wait)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 Wait for Trigger (Halt before Work Unit)                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14:12   | FLOW     | Next Operation.<br>The DMA_CFG.FLOW selects descriptor handling options.                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 0<br>STOP - Stop<br>When the current work unit completes, the DMA channel stops automatically, after signaling an interrupt (if selected). The DMA_STAT.RUN status bit changes to idle, while DMA_CFG.EN bit is unchanged. In this state, the channel is stopped. Peripheral interrupts are still filtered out by the DMA. The channel may be restarted simply by another write (with the DMA_CFG.EN set) to the DMA_CFG register specifying the next work unit.                                  |
|         |          | 1<br>AUTO - Autobuffer<br>In this mode, no descriptors in memory are used. Instead, DMA is performed in a continuous circular buffer fashion based on user programmed DMA MMR settings. On completion of the work unit, the parameter registers are reloaded into the current registers, and DMA resumes immediately with zero overhead. This mode is considered to be a succession of automatically restarted work units. Autobuffer mode is stopped by a user write of 0 to the DMA_CFG.EN bit. |
|         |          | 2<br>Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 3<br>Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 4<br>DSCL - Descriptor List<br>This mode fetches a descriptor Set from memory that includes DMA_DSCPTR_NXT, allowing maximum flexibility in locating descriptors in memory.                                                                                                                                                                                                                                                                                                                       |
|         |          | 5<br>DSCA - Descriptor Array<br>This mode fetches a descriptor set from memory that does not include the DMA_DSCPTR_NXT element. Because the descriptor set does not contain a next descriptor pointer entry, the DMA defaults to using the DMA_DSCPTR_CUR register to step through descriptors, allowing a group of descriptors sets to follow one another in                                                                                                                                    |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10:8    | MSIZE    | <p>Memory Transfer Word Size.</p> <p>The DMA_CFG.MSIZE bits select memory transfer sizes of 8-, 16-, 32-, 64-, 128-, or 256-bit words. Note that the transfer start address (DMA_ADDRSTART) and transfer increment values (DMA_XMOD, and if needed DMA_YMOD) must be a multiple of the memory transfer unit size.</p>                                                                                                                                                                              |
|         |          | 0 1 Byte                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 2 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 4 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 3 8 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 4 16 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 5 32 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 6:4     | PSIZE    | <p>Peripheral Transfer Word Size.</p> <p>The DMA_CFG.PSIZE bits select peripheral transfer sizes as 8, 16, 32, or 64 bits wide. Each request/grant results in a single peripheral access. There is no bursting on the peripheral bus, so the DMA_CFG.PSIZE selection must be less than, or equal to, the width of the bus. If the selection is greater than the bus width, a configuration error occurs. Note that the processor's peripheral bus is dedicated to DMA and peripheral accesses.</p> |
|         |          | 0 1 Byte                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 2 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 4 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 3 8 Bytes                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                       |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 3       | CADDR    | Use Current Address.<br>If the DMA_CFG.CADDR bit is cleared, the DMA loads the DMA_ADDRSTART register on the first access of the work unit. If the DMA_CFG.CADDR bit is set, the DMA uses the DMA_ADDR_CUR register value for the starting address for the work unit and writes the same value to the DMA_ADDRSTART register.<br>This operation permits continuation of a previous work unit. If this mode is used at the end of a descriptor list or array, the DMA ignores the start address value that is fetched as part of the descriptor set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Load Starting Address |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Use Current Address   |
| 2       | SYNC     | Synchronize Work Unit Transitions.<br>Setting the DMA_CFG.SYNC bit clears the DMA FIFO and pointers before starting the first Work Unit of a Work Unit Chain.<br>When the transfer direction is memory read/transmit (DMA_CFG.WNR =0), the DMA waits until all data has been transmitted to peripheral before moving on to next Work Unit, clearing the FIFO and pointers.<br>When the transfer direction is memory write/receive (DMA_CFG.WNR =1), the DMA ignores the DMA_CFG.SYNC bit value after processing the first Work Unit of a Work Unit Chain. Because the channel is allowed to receive data when turned on but idling, there could be data in the FIFO that was put in by the peripheral before the channel was programmed. With DMA_CFG.SYNC set at the beginning of a work unit chain (during the first work unit), the DMA clears the FIFO, erasing the data put in to the FIFO while the channel was idling.<br>Syncing lets you change the DMA_CFG.PSIZE between individual work units and (in some cases) work unit chains. The sync resets the pointers in the FIFO, preventing misaligned FIFO access.<br>The DMA_CFG.MSIZE may be changed between consecutive work units, independent of the DMA_CFG.SYNC bit setting.<br>Syncing also permits changes to transfer direction. And, because the data in the FIFO is eliminated, the data that went into the FIFO from one direction (transmit or receive) is not sent back in the other direction after the direction change. |                       |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | No Synchronization    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Synchronize Channel   |

**Table 13-18:** DMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | WNR      | Write/Read Channel Direction.<br>The DMA_CFG.WNR selects receive (write to memory) or transmit (read from memory) channel direction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0 Transmit (Read from memory)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Receive (Write to memory)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 0       | EN       | DMA Channel Enable.<br>The DMA_CFG.EN enables the selected DMA Channel.<br>When a peripheral DMA channel is enabled, data requests from the peripheral denote DMA requests. When a channel is disabled, the DMA unit ignores the peripheral data request and passes it directly to the system event controller.<br>To avoid unexpected results, take care to enable the DMA channel before enabling the peripheral, and to disable the peripheral before disabling the DMA channel.<br>A transition of DMA_CFG.EN from 0 to 1 creates a hard reset of all internal counters and state, including the DMA_STAT register. (All other register values remain unaffected.) A transition from 1 to 0 maintains all counters and registers for the user to read and analyze. If a descriptor is loaded (see DMA_CFG.FLOW field) with DMA_CFG.EN cleared, the DMA goes to off/idle state after the descriptor load is complete. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

## Inner Loop Count Start Value

For 2D DMA, the DMA\_XCNT contains the inner loop count. This value selects the number of DMA\_CFG.MSIZE size data transfers to make up the length of a row. For 1D DMA, DMA\_XCNT specifies the number of DMA\_CFG.MSIZE size data transfers for the entire work unit. The DMA\_XCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if DMA\_XCNT is 0x0 when a work unit begins.

DMA\_XCNT: Inner Loop Count Start Value - R/W

Reset = 0x0000 0000

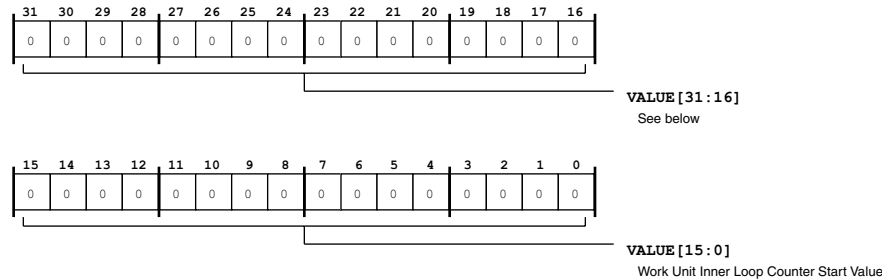


Figure 13-7: DMA\_XCNT Register Diagram

Table 13-19: DMA\_XCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                   |
|---------|----------|-------------------------------------------|
| 31:0    | VALUE    | Work Unit Inner Loop Counter Start Value. |

Inner Loop Address Increment

The DMA\_XMOD contains a signed, two's-complement byte address increment. In 1D DMA, this increment is the stride that is applied after each DMA\_CFG.MSIZE size data transfer. The DMA\_XMOD register is read/write prior to enabling the channel, but is read-only after enabling channel.

The DMA\_XMOD value is specified in bytes, regardless of the work unit size. In 2D DMA, this increment is applied after each DMA\_CFG.MSIZE size data transfer in the inner loop, up to but not including the last DMA\_CFG.MSIZE size data transfer in each inner loop. After the last DMA\_CFG.MSIZE size data transfer in each inner loop, the DMA\_YMOD register is applied instead, including the last DMA\_CFG.MSIZE size data transfer of a work unit.

The DMA\_XMOD field may be set to 0. In this case, DMA is performed repeatedly to or from the same address. This approach can be useful for transferring data between a data register and an external memory-mapped peripheral.

DMA\_XMOD: Inner Loop Address Increment - R/W

Reset = 0x0000 0000

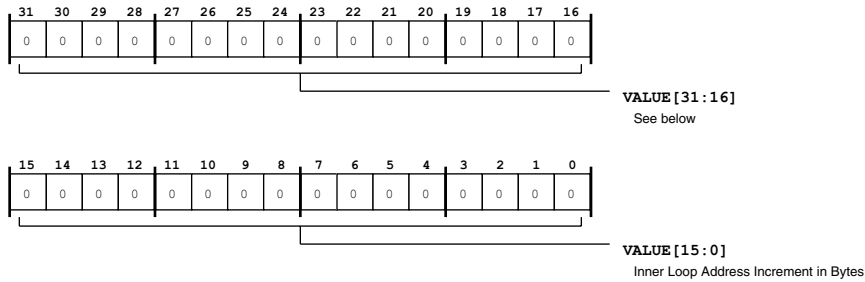


Figure 13-8: DMA\_XMOD Register Diagram

Table 13-20: DMA\_XMOD Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 31:0    | VALUE    | Inner Loop Address Increment in Bytes. |

Outer Loop Count Start Value (2D only)

For 2D DMA, the DMA\_YCNT contains the outer loop count. This register is not used in 1D DMA mode. The DMA\_YCNT register specifies the number of rows in the outer loop of a 2D DMA sequence. The DMA\_YCNT register is read/write prior to enabling the channel, but is read-only after enabling channel. Note that the DMA generates a configuration error if DMA\_YCNT is 0x0 when a work unit begins.

DMA\_YCNT: Outer Loop Count Start Value (2D only) - R/W

Reset = 0x0000 0000

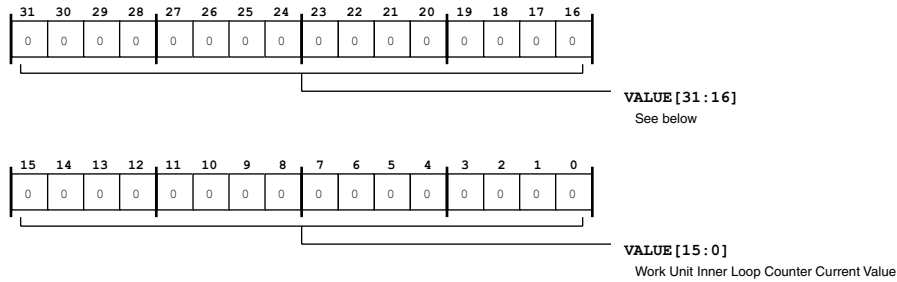


Figure 13-9: DMA\_YCNT Register Diagram



**Table 13-21:** DMA\_YCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                     |
|---------|----------|---------------------------------------------|
| 31:0    | VALUE    | Work Unit Inner Loop Counter Current Value. |

## Outer Loop Address Increment (2D only)

The DMA\_YMOD contains a signed, two's-complement value. This byte address increment is applied after each decrement of the DMA\_YCNT\_CUR register. The value is the offset between the last word of one row and the first word of the next row. Note that DMA\_YMOD is specified in bytes, regardless of the work unit size. The DMA\_YMOD register is read/write prior to enabling the channel, but is read-only after enabling channel.

**DMA\_YMOD: Outer Loop Address Increment (2D only) - R/W**

Reset = 0x0000 0000

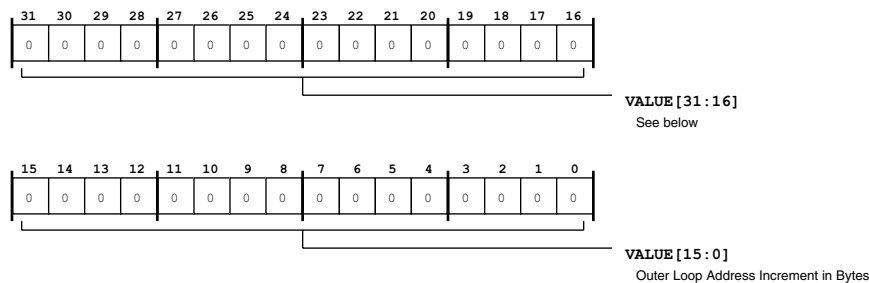


**Figure 13-10:** DMA\_YMOD Register Diagram

**Table 13-22:** DMA\_YMOD Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 31:0    | VALUE    | Outer Loop Address Increment in Bytes. |

## Current Descriptor Pointer

The DMA\_DSCPTR\_CUR contains the memory address for the next descriptor to be loaded. The DMA\_DSCPTR\_CUR register is read/write prior to enabling the channel, but is read-only after enabling channel. For DMA\_CFG.FLOW mode settings that involve descriptor fetches, this register is used to read descriptors into appropriate MMRs before a work unit begins. For descriptor list mode, the DMA\_DSCPTR\_CUR is initialized from the DMA\_DSCPTR\_NXT register before fetching each descriptor set. Then, the address in the DMA\_DSCPTR\_CUR register increments as each descriptor is read in.

When the entire descriptor set has been read, the DMA\_DSCPTR\_CUR register contains this value:

DMA\_DSCPTR\_CUR = Descriptor Start Address + Descriptor Size (# of elements)

For descriptor array mode, the DMA\_DSCPTR\_CUR register, and not the DMA\_DSCPTR\_NXT register, must be programmed by MMR access before starting DMA operation.

DMA\_DSCPTR\_CUR: Current Descriptor Pointer - R/W

Reset = 0x0000 0000

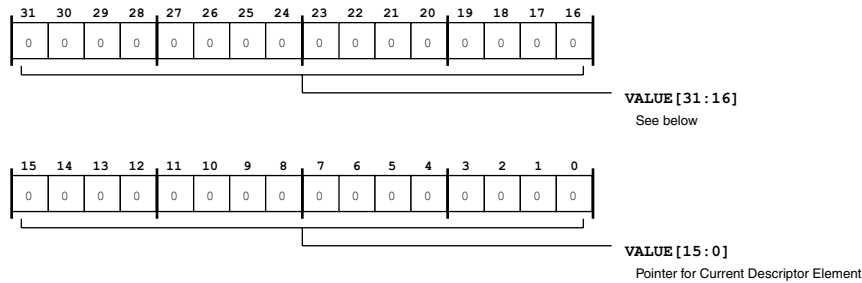


Figure 13-11: DMA\_DSCPTR\_CUR Register Diagram

Table 13-23: DMA\_DSCPTR\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                 |
|---------|----------|-----------------------------------------|
| 31:0    | VALUE    | Pointer for Current Descriptor Element. |

Previous Initial Descriptor Pointer

The DMA\_DSCPTR\_PRV contains the initial descriptor pointer for the previous work unit. If DMA\_CFG.DESIDCPY is set, the DMA copies the initial descriptor pointer to DMA\_DSCPTR\_PRV after the work unit completes. Otherwise, the value is not updated.

To indicate an overrun, bit 0 of DMA\_DSCPTR\_PRV is used as a previous descriptor pointer overrun (PDPO) status bit. Due to aligned addressing combined with all descriptors being 32 bits in width, bits 0 and 1 of all descriptor pointers must be zero. Otherwise, an alignment error would occur when used for descriptor fetches. As a result, bit 1 and 0 of DMA\_DSCPTR\_PRV may be used for status. For more information, see the section on descriptor pointer capture in the DMA functional description.

DMA\_DSCPTR\_PRV: Previous Initial Descriptor Pointer - R/NW

Reset = 0x0000 0000

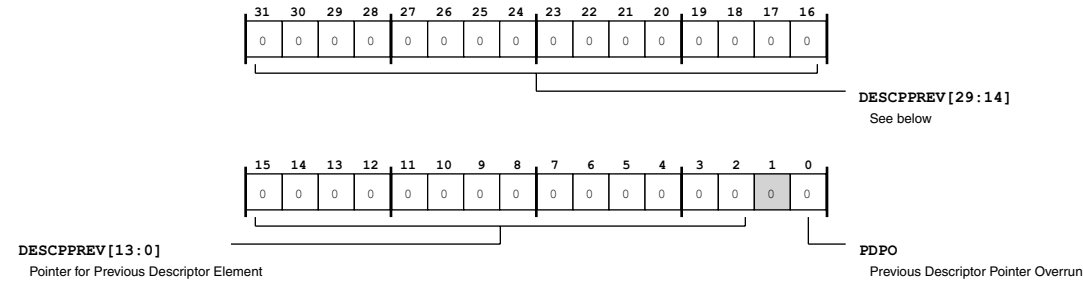


Figure 13-12: DMA\_DSCPTR\_PRV Register Diagram

Table 13-24: DMA\_DSCPTR\_PRV Register Fields

| Bit No. | Bit Name  | Description/Enumeration                  |
|---------|-----------|------------------------------------------|
| 31:2    | DESCPPREV | Pointer for Previous Descriptor Element. |
| 0       | PDPO      | Previous Descriptor Pointer Overrun.     |

Current Address

The DMA\_ADDR\_CUR contains the present memory transfer address for a given work unit. At the start of a work unit, the DMA\_ADDR\_CUR is loaded from the DMA\_ADDRSTART register, and DMA\_ADDR\_CUR is incremented as each transfer occurs. The DMA\_ADDR\_CUR register is read/write prior to enabling the channel, but is read-only after enabling channel.

DMA\_ADDR\_CUR: Current Address - R/W

Reset = 0x0000 0000

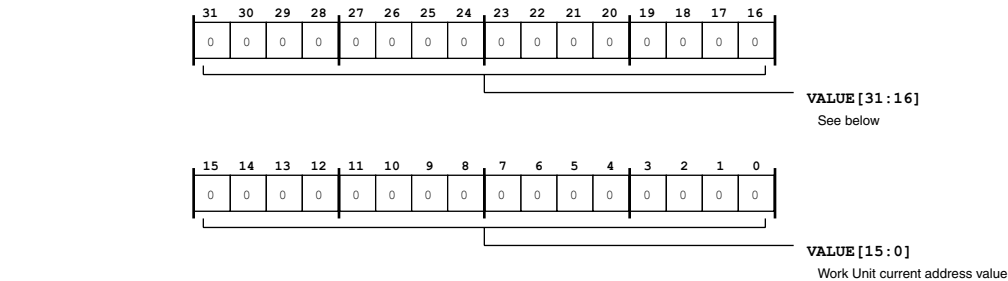


Figure 13-13: DMA\_ADDR\_CUR Register Diagram

Table 13-25: DMA\_ADDR\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 31:0    | VALUE    | Work Unit current address value. |

Status Register

The DMA\_STAT indicates status of DMA work units, FIFO, errors, interrupts, and triggers.

DMA\_STAT: Status Register - R/W

Reset = 0x0000 6000

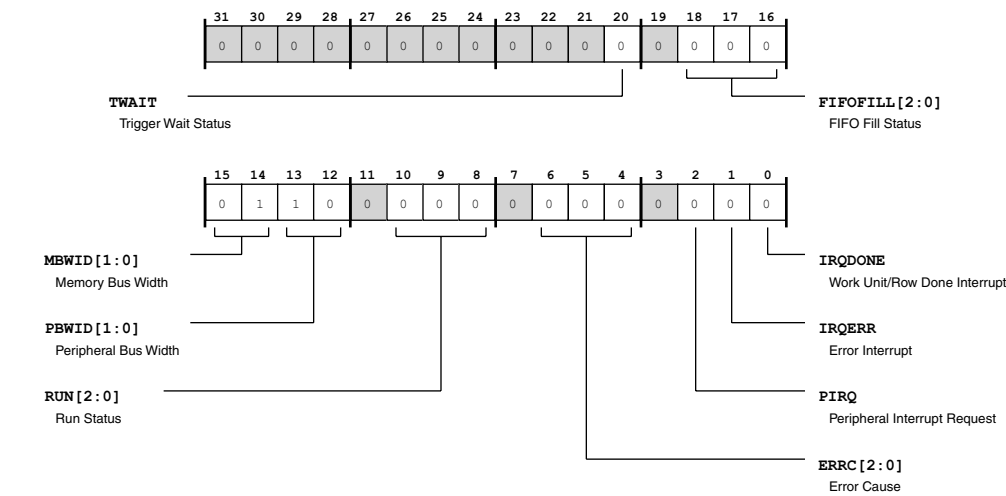


Figure 13-14: DMA\_STAT Register Diagram

Table 13-26: DMA\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | TWAIT    | Trigger Wait Status.<br>The DMA_STAT.TWAIT indicates whether the DMA has or has not received a trigger. This bit is set until it reaches the next wait state. At that point, the bit is cleared, the DMA stops processing that work unit, and the following work unit is processed. The DMA does not distinguish between one or more triggers received. |
|         |          | 0 No trigger received                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Trigger received                                                                                                                                                                                                                                                                                                                                      |

**Table 13-26:** DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                        |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | FIFO FILL | FIFO Fill Status.<br>The <code>DMA_STAT.FIFO FILL</code> reports the quantity of data in the FIFO relative to available space. |
|         |           | 0 Empty                                                                                                                        |
|         |           | 1 Empty < FIFO = 1/4 Full                                                                                                      |
|         |           | 2 1/4 Full < FIFO = 1/2 Full                                                                                                   |
|         |           | 3 1/2 Full < FIFO = 3/4 Full                                                                                                   |
|         |           | 4 3/4 Full < FIFO = Full                                                                                                       |
|         |           | 5 Reserved                                                                                                                     |
|         |           | 6 Reserved                                                                                                                     |
|         |           | 7 Full                                                                                                                         |
| 15:14   | MBWID     | Memory Bus Width.<br>The <code>DMA_STAT.MBWID</code> indicates the width of the memory bus connected to this DMA.              |
|         |           | 0 2 Bytes                                                                                                                      |
|         |           | 1 4 Bytes                                                                                                                      |
|         |           | 2 8 Bytes                                                                                                                      |
|         |           | 3 16 Bytes                                                                                                                     |
| 13:12   | PBWID     | Peripheral Bus Width.<br>The <code>DMA_STAT.PBWID</code> indicates the width of the peripheral bus connected to this DMA.      |
|         |           | 0 1 Byte                                                                                                                       |
|         |           | 1 2 Bytes                                                                                                                      |
|         |           | 2 4 Bytes                                                                                                                      |
|         |           | 3 8 Bytes                                                                                                                      |

**Table 13-26:** DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10:8    | RUN      | <p>Run Status.</p> <p>The <code>DMA_STAT.RUN</code> reports the DMA's current operational state. If the DMA is in idle or stop state, the <code>DMA_CFG.EN</code> bit may be either 0 or 1. Note that the <code>DMA_STAT.RUN</code> is not cleared by a transition of the <code>DMA_CFG.EN</code> bit from 0 to 1. The <code>DMA_STAT.RUN</code> is automatically cleared when the DMA completes.</p> |
|         |          | 0 Idle/Stop State                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Descriptor Fetch                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 2 Data Transfer                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 3 Waiting for Trigger                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 4 Waiting for Write ACK/FIFO Drain to Peripheral                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 5 Reserved                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 6 Reserved                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 7 Reserved                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 13-26:** DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:4     | ERRC     | Error Cause.<br>When an interrupt request error occurs (DMA_STAT.IRQERR), the DMA updates DMA_STAT.ERRC to identify the type of error. For more information, see the errors section of the DMA functional description.                                                                                                                        |
|         |          | 0 Configuration Error                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Illegal Write Occurred While Channel Running                                                                                                                                                                                                                                                                                                |
|         |          | 2 Address Alignment Error                                                                                                                                                                                                                                                                                                                     |
|         |          | 3 Memory Access/Fabric Error                                                                                                                                                                                                                                                                                                                  |
|         |          | 4 Reserved                                                                                                                                                                                                                                                                                                                                    |
|         |          | 5 Trigger Overrun                                                                                                                                                                                                                                                                                                                             |
|         |          | 6 Bandwidth Monitor Error                                                                                                                                                                                                                                                                                                                     |
| 2       | PIRQ     | Peripheral Interrupt Request.<br>The DMA_STAT.PIRQ indicates an interrupt has been caused by the peripheral. Programmers can use the DMA_STAT.PIRQ status to help determine which DMA asserted the interrupt and to help distinguish between an interrupt caused based on the state of the work unit and an interrupt made by the peripheral. |
|         |          | 0 No Interrupt                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 Interrupt Signaled by Peripheral                                                                                                                                                                                                                                                                                                            |

**Table 13-26:** DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 1       | IRQERR   | Error Interrupt.<br>The DMA_STAT . IRQERR indicates that the DMA has detected a documented rule violations during DMA programming or operation. The DMA cannot, however, flag all possible programming or operation issues to indicate errors. Programmers can use DMA_STAT . IRQERR to help determine which DMA issued the error interrupt. Note that the DMA_STAT . IRQERR is not cleared by a transition of the DMA_CFG . EN bit from 0 to 1. The DMA_STAT . IRQERR must be cleared with a write-1-to-clear operation prior to the DMA_CFG . EN transition for the fields to be reset. |                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Error       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Error Occurred |
| 0       | IRQDONE  | Work Unit/Row Done Interrupt.<br>The DMA_STAT . IRQDONE indicates the DMA has detected the completion of a work unit or row (inner loop count) and has issued an interrupt. Programmers can use the DMA_STAT . IRQDONE status to help determine which DMA asserted the interrupt and to help distinguish between an interrupt caused based on the state of the work unit and an interrupt made by the peripheral. For more information, see the interrupts section of the DMA functional description.                                                                                     |                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Inactive       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Active         |

## Current Count(1D) or intra-row XCNT (2D)

For 1D DMA, the DMA loads the DMA\_XCNT\_CUR from the DMA\_XCNT register at the beginning of each work unit. For 2D DMA, the DMA loads DMA\_XCNT\_CUR from the DMA\_XCNT register after the end of each row. The DMA decrements the value in DMA\_XCNT\_CUR each time a DMA\_CFG . MSIZE size data transfer occurs. When the count in DMA\_XCNT\_CUR expires, the work unit is complete. In 2D DMA, the DMA\_XCNT\_CUR value is 0 only when the entire transfer is complete.



DMA\_XCNT\_CUR: Current Count(1D) or intra-row XCNT (2D) - R/NW

Reset = 0x0000 0000

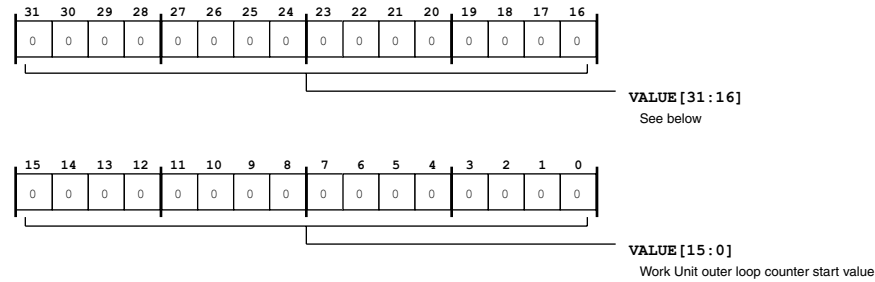


Figure 13-15: DMA\_XCNT\_CUR Register Diagram

Table 13-27: DMA\_XCNT\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                   |
|---------|----------|-------------------------------------------|
| 31:0    | VALUE    | Work Unit outer loop counter start value. |

Current Row Count (2D only)

For 2D DMA, the DMA loads the DMA\_YCNT\_CUR from the DMA\_YCNT register at the beginning of each 2D DMA session. The DMA\_YCNT\_CUR is not used for 1D DMA. The DMA decrements DMA\_YCNT\_CUR each time the DMA\_XCNT\_CUR expires during 2D DMA operation, signifying completion of an entire row transfer.

DMA\_YCNT\_CUR: Current Row Count (2D only) - R/NW

Reset = 0x0000 0000

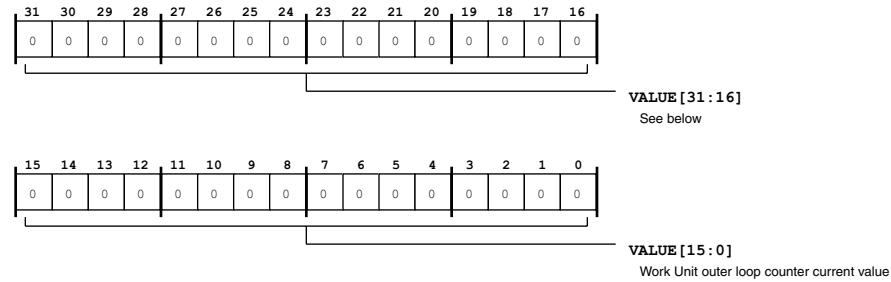


Figure 13-16: DMA\_YCNT\_CUR Register Diagram

Table 13-28: DMA\_YCNT\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                     |
|---------|----------|---------------------------------------------|
| 31:0    | VALUE    | Work Unit outer loop counter current value. |

Bandwidth Limit Count

The DMA\_BWLCNT contains a count that determines how often the DMA issues memory transactions. The DMA loads the value from DMA\_BWLCNT into DMA\_BWLCNT\_CUR and decrements the current value each SCLK cycle. When DMA\_BWLCNT\_CUR reaches 0x0000, the next request is issued, and the DMA reloads DMA\_BWLCNT\_CUR. This bandwidth limit functionality is not applied to descriptor fetch requests. Programming 0x0000 allows the DMA to request as often as possible. 0xFFFF is a special case and causes requests to stop.

DMA\_BWLCNT: Bandwidth Limit Count - R/W

Reset = 0x0000 0000

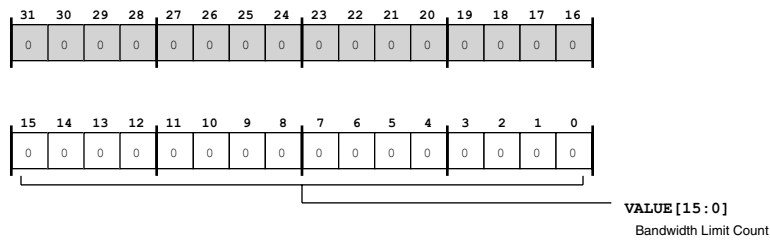


Figure 13-17: DMA\_BWLCNT Register Diagram

Table 13-29: DMA\_BWLCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Bandwidth Limit Count.  |

Bandwidth Limit Count Current

The DMA\_BWLCNT\_CUR contains the number of SCLK count cycles remaining before the next request is issued.

**DMA\_BWLCNT\_CUR: Bandwidth Limit Count Current - R/NW**

Reset = 0x0000 0000

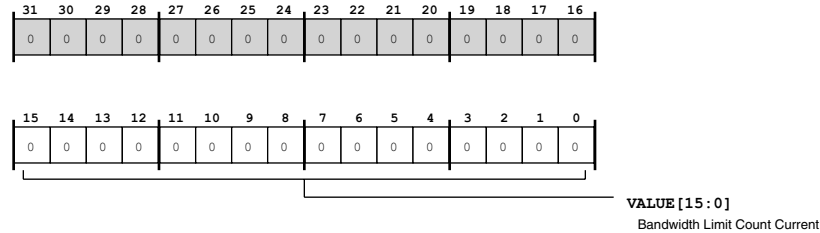


**Figure 13-18:** DMA\_BWLCNT\_CUR Register Diagram

**Table 13-30:** DMA\_BWLCNT\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration        |
|---------|----------|--------------------------------|
| 15:0    | VALUE    | Bandwidth Limit Count Current. |

## Bandwidth Monitor Count

The DMA\_BWMCNT contains the maximum number of SCLK cycles allowed for a work unit to complete. Each time the DMA\_CFG register is written (MMR access only), a work unit ends, or an autobuffer wraps, the DMA loads the value in DMA\_BWMCNT into DMA\_BWMCNT\_CUR. The DMA decrements DMA\_BWMCNT\_CUR every SCLK a work unit is active. If DMA\_BWMCNT\_CUR reaches 0x0000\_0000, the DMA\_STAT . IRQERR bit is set, and the DMA\_STAT . ERRRC is set to 0x6. The DMA\_BWMCNT\_CUR remains at 0x0000\_0000 until it is reloaded when the work unit completes. Unlike other error causes, a bandwidth monitor error does not stop work unit processing. Programming 0x0000\_0000 disables bandwidth monitor functionality.

**DMA\_BWMCNT: Bandwidth Monitor Count - R/W**

Reset = 0x0000 0000

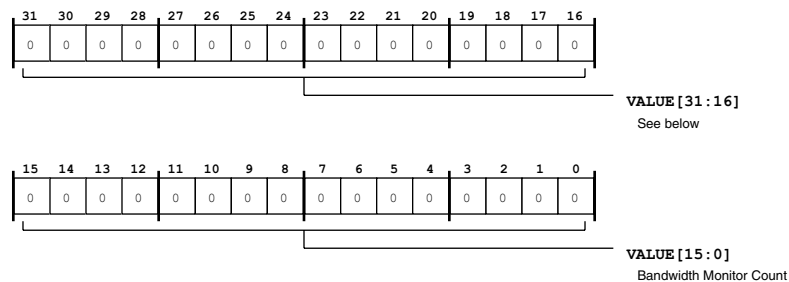


**Figure 13-19:** DMA\_BWMCNT Register Diagram

**Table 13-31:** DMA\_BWMCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Bandwidth Monitor Count. |

## Bandwidth Monitor Count Current

The DMA\_BWMCNT\_CUR contains the number of cycles remaining for the current descriptor to complete.

**DMA\_BWMCNT\_CUR: Bandwidth Monitor Count Current - R/NW**

Reset = 0x0000 0000

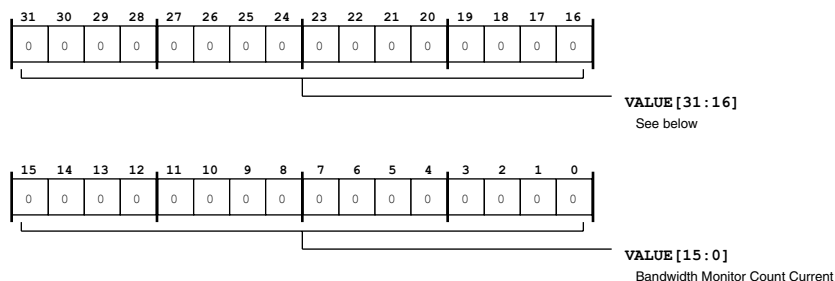


**Figure 13-20:** DMA\_BWMCNT\_CUR Register Diagram

**Table 13-32:** DMA\_BWMCNT\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 31:0    | VALUE    | Bandwidth Monitor Count Current. |

## DMA Channel List for ADSP-BF60x

The following tables provide DMA channel assignment and channel parametric information for the ADSP-BF60x processors.

**Table 13-33:** SPORT DMA Channels

| DMA Channel | Peripheral       | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA0        | SPORT0 Channel A | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-33:** SPORT DMA Channels (Continued)

| DMA Channel | Peripheral       | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA1        | SPORT0 Channel B | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA2        | SPORT1 Channel A | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA3        | SPORT1 Channel B | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA4        | SPORT2 Channel A | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA5        | SPORT2 Channel B | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-34:** SPI DMA Channels

| DMA Channel | Peripheral    | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|---------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA6        | SPI0 Transmit | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA7        | SPI0 Receive  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA8        | SPI1 Transmit | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA9        | SPI1 Receive  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-35:** RSI DMA Channels

| DMA Channel | Peripheral | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA10       | RSI0       | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-36:** SDU DMA Channels

| DMA Channel | Peripheral | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA11       | SDU0       | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-37:** Link Port DMA Channels

| DMA Channel | Peripheral | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA13       | Linkport0  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA14       | Linkport1  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA15       | Linkport2  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |
| DMA16       | Linkport3  | 64                 | No                               | 32 bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-38:** UART DMA Channels

| DMA Channel | Peripheral     | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|----------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA17       | UART0 Transmit | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA18       | UART0 Receive  | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA19       | UART1 Transmit | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |
| DMA20       | UART1 Receive  | 64                 | No                               | 32-bit                             | 32-bit                                 | 4                     | 4                      |

**Table 13-39:** MDMA Channels

| DMA Channel | Peripheral                | FIFO Depth (Bytes) | Bandwidth Limit/Monit or Support | Memory Bus Width (DMA_STAT .MSIZE) | Peripheral Bus Width (DMA_STAT .PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|---------------------------|--------------------|----------------------------------|------------------------------------|----------------------------------------|-----------------------|------------------------|
| DMA21       | MDMA0 Source/ Destination | 128                | Yes                              | 32-bit                             | 32-bit                                 | 8                     | 7                      |
| DMA22       | MDMA0 Destination/Source  | 64                 | Yes                              | 32-bit                             | 32-bit                                 | 8                     | 4                      |
| DMA23       | MDMA1 Source/ Destination | 64                 | Yes                              | 32-bit                             | 32-bit                                 | 8                     | 4                      |
| DMA24       | MDMA1 Destination/Source  | 64                 | Yes                              | 32-bit                             | 32-bit                                 | 8                     | 4                      |

**Table 13-39:** MDMA Channels (Continued)

| <b>DMA Channel</b> | <b>Peripheral</b>        | <b>FIFO Depth (Bytes)</b> | <b>Bandwidth Limit/Monitor or Support</b> | <b>Memory Bus Width (DMA_STAT.MSIZE)</b> | <b>Peripheral Bus Width (DMA_STAT.PSIZE)</b> | <b>Max Outstanding Reads</b> | <b>Max Outstanding Writes</b> |
|--------------------|--------------------------|---------------------------|-------------------------------------------|------------------------------------------|----------------------------------------------|------------------------------|-------------------------------|
| DMA25              | MDMA2 Source/Destination | 128                       | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA26              | MDMA2 Destination/Source | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |
| DMA27              | MDMA3 Source/Destination | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |
| DMA28              | MDMA3 Destination/Source | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |

**Table 13-40:** CRC DMA Channels

| <b>DMA Channel</b> | <b>Peripheral</b> | <b>FIFO Depth (Bytes)</b> | <b>Bandwidth Limit/Monitor or Support</b> | <b>Memory Bus Width (DMA_STAT.MSIZE)</b> | <b>Peripheral Bus Width (DMA_STAT.PSIZE)</b> | <b>Max Outstanding Reads</b> | <b>Max Outstanding Writes</b> |
|--------------------|-------------------|---------------------------|-------------------------------------------|------------------------------------------|----------------------------------------------|------------------------------|-------------------------------|
| DMA21              | CRC0 Receive      | 128                       | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA22              | CRC0 Transmit     | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |
| DMA23              | CRC1 Receive      | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |
| DMA24              | CRC1 Transmit     | 64                        | Yes                                       | 32-bit                                   | 32-bit                                       | 8                            | 4                             |



**Table 13-41:** EPPI DMA Channels

| <b>DMA Channel</b> | <b>Peripheral</b>          | <b>FIFO Depth (Bytes)</b> | <b>Bandwidth Limit/Monitor Support</b> | <b>Memory Bus Width (DMA_STAT.MSIZE)</b> | <b>Peripheral Bus Width (DMA_STAT.PSIZE)</b> | <b>Max Outstanding Reads</b> | <b>Max Outstanding Writes</b> |
|--------------------|----------------------------|---------------------------|----------------------------------------|------------------------------------------|----------------------------------------------|------------------------------|-------------------------------|
| DMA29              | EPPI0 Luma/Pixel Pipe Data | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA30              | EPPI0 Chrominance Data     | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA31              | EPPI2 Luma/Pixel Pipe Data | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA32              | EPPI2 Chrominance Data     | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA33              | EPPI1 Luma/Pixel Pipe Data | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA34              | EPPI1 Chrominance Data     | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |

**Table 13-42:** PIXC DMA Channels

| <b>DMA Channel</b> | <b>Peripheral</b>    | <b>FIFO Depth (Bytes)</b> | <b>Bandwidth Limit/Monitor Support</b> | <b>Memory Bus Width (DMA_STAT.MSIZE)</b> | <b>Peripheral Bus Width (DMA_STAT.PSIZE)</b> | <b>Max Outstanding Reads</b> | <b>Max Outstanding Writes</b> |
|--------------------|----------------------|---------------------------|----------------------------------------|------------------------------------------|----------------------------------------------|------------------------------|-------------------------------|
| DMA35              | PIXC Input Channel A | 128                       | Yes                                    | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA36              | PIXC Input Channel B | 128                       | Yes                                    | 32-bit                                   | 32-bit                                       | 8                            | 7                             |

**Table 13-42:** PIXC DMA Channels (Continued)

| DMA Channel | Peripheral  | FIFO Depth (Bytes) | Bandwidth Limit/Monitor Support | Memory Bus Width (DMA_STAT.MSIZE) | Peripheral Bus Width (DMA_STAT.PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|-------------|--------------------|---------------------------------|-----------------------------------|---------------------------------------|-----------------------|------------------------|
| DMA37       | PIXC Output | 128                | Yes                             | 32-bit                            | 32-bit                                | 8                     | 7                      |

**Table 13-43:** PVP DMA Channels

| DMA Channel | Peripheral                           | FIFO Depth (Bytes) | Bandwidth Limit/Monitor Support | Memory Bus Width (DMA_STAT.MSIZE) | Peripheral Bus Width (DMA_STAT.PSIZE) | Max Outstanding Reads | Max Outstanding Writes |
|-------------|--------------------------------------|--------------------|---------------------------------|-----------------------------------|---------------------------------------|-----------------------|------------------------|
| DMA38       | PVP0 Camera Pipe Data Output Port 1  | 128                | No                              | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA39       | PVP0 Camera Pipe Data Output Port 2  | 128                | No                              | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA40       | PVP0 Camera Pipe Status Output       | 64                 | No                              | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA41       | PVP0 Camera Pipe Configuration Input | 64                 | Yes                             | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA42       | PVP0 Memory Pipe Data Output         | 128                | Yes                             | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA43       | PVP0 Memory Pipe Data Input          | 128                | Yes                             | 32-bit                            | 32-bit                                | 8                     | 7                      |
| DMA44       | PVP0 Memory Pipe Status Output       | 64                 | No                              | 32-bit                            | 32-bit                                | 8                     | 7                      |

**Table 13-43:** PVP DMA Channels (Continued)

| <b>DMA Channel</b> | <b>Peripheral</b>                    | <b>FIFO Depth (Bytes)</b> | <b>Bandwidth Limit/Monitor Support</b> | <b>Memory Bus Width (DMA_STAT.MSIZE)</b> | <b>Peripheral Bus Width (DMA_STAT.PSIZE)</b> | <b>Max Outstanding Reads</b> | <b>Max Outstanding Writes</b> |
|--------------------|--------------------------------------|---------------------------|----------------------------------------|------------------------------------------|----------------------------------------------|------------------------------|-------------------------------|
| DMA45              | PVP0 Memory Pipe Configuration Input | 64                        | Yes                                    | 32-bit                                   | 32-bit                                       | 8                            | 7                             |
| DMA46              | PVP0 Camera Pipe Data Output Port 0  | 128                       | No                                     | 32-bit                                   | 32-bit                                       | 8                            | 7                             |



# 14 General-Purpose Ports (PORT)

This section describes general-purpose ports, pin multiplexing, general-purpose input/output (GPIO) functionality, and pin interrupts.

The general-purpose ports provide the following three functions.

- Pin multiplexing scheme
- GPIO functionality
- Pin interrupts

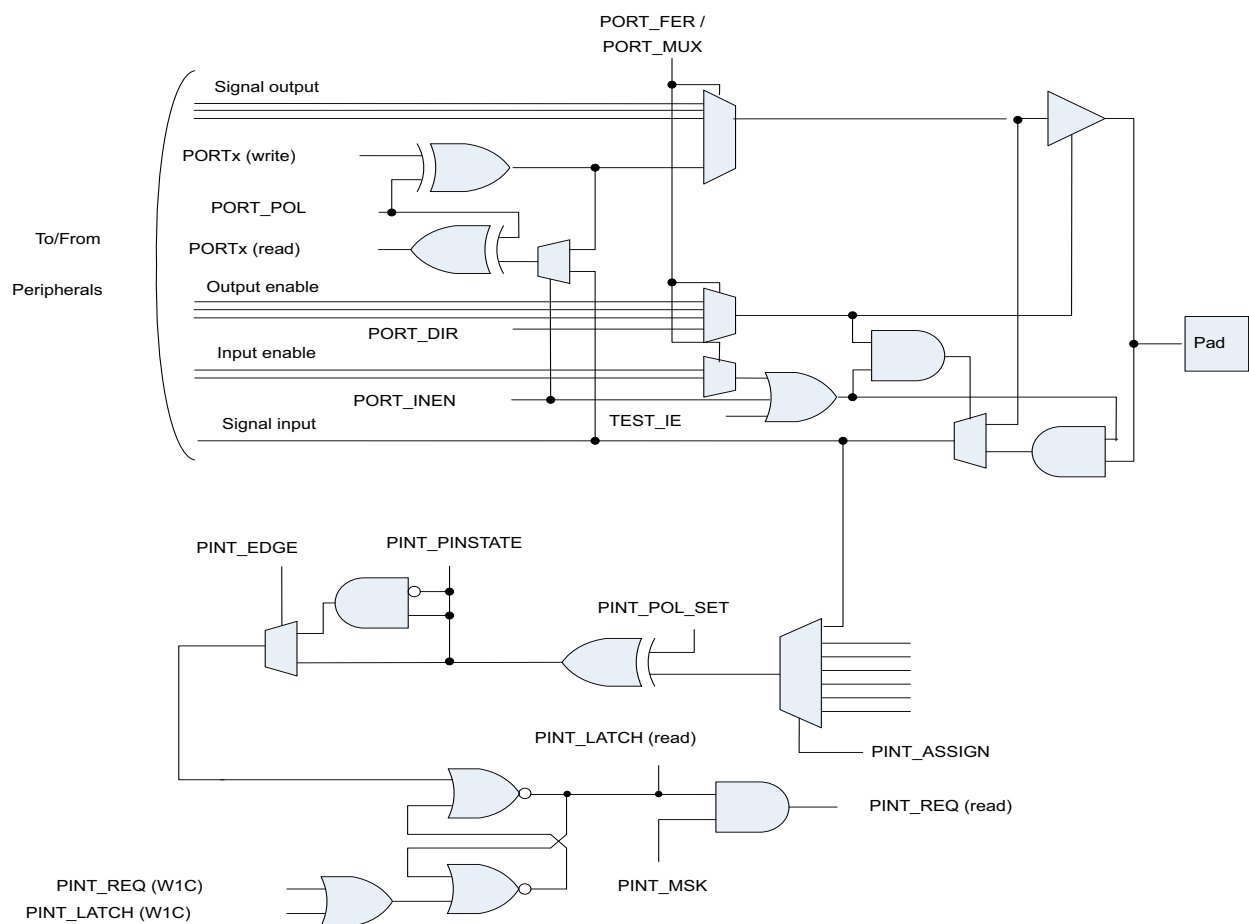


**Figure 14-1:** Simplified GPIO and Pin Interrupt Signal Flow

## PORT Features

The PORTs include the following features:

- Up to 112 general-purpose I/O (GPIO) pins
- Input mode, output mode, and open-drain mode of GPIO operation
- Port multiplexing controlled by individual pin-per-pin base
- No glue hardware required for unused pins
- Six interrupt channels dedicated to pin interrupts
- All port pins provide interrupt functionality
- Byte-wide pin-to-interrupt assignment

## PORT Functional Description

Every port pin can operate in GPIO mode. This is the default after reset and is controlled by the port-specific `PORTx_FER` enable register. Every port has a dedicated set of MMR registers that control the GPIO functionality. Every bit in these registers represents a certain GPIO pin of the specific port.

By default, every GPIO is set to input mode. The input drivers are not enabled, which avoids the need for unnecessary current sinks and the external pulling of resistors on unused or *do not care* pins.

### Input Mode

The default mode of every GPIO pin after reset is input mode, but the input drivers are not enabled. To enable any GPIO input drivers, set the corresponding bits in the input enable register `PORTx_INEN`. When enabled, a read from the `PORTx` register returns the logical state of the input pin. The input signal does not overwrite the state of the flip-flop used for the output case. That state can only be altered by software. If the input driver is enabled, a write to the `PORTx` register can alter the state of the flip-flop, but the change cannot be read back.

### Output Mode

Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the corresponding bits in the direction registers. Direction registers are implemented as a pair of write-1-to-set (W1S) and write-1-to-clear (W1C) MMRs, called `PORTx_DIR_SET` and `PORTx_DIR_CLEAR`. This way, the direction of the signal flow of individual GPIO pins can be altered by separate software threads without mutually impacting other GPIOs on the same port.

Both `PORTx_DIR_SET` and `PORTx_DIR_CLEAR` registers return the same value when read and a logical 1 indicates an enabled output. The state of output pins is controlled by the `PORTx` registers. A logical 0 drives the output low while a logical 1 drives the output high.

While the `PORTx` register can be written to alter all GPIOs of a specific port at once, there is also a pair of `W1S` and `W1C` MMRs, called `PORTx_SET` and `PORTx_CLEAR` that enable manipulation of individual GPIO outputs. The state of the outputs can be obtained by reading the `PORTx` registers. Because the state of the GPIO output can already be controlled before the output driver is enabled, it is recommended to first set or clear the flip-flop to avoid any volatile levels on the output.

## Open-Drain Mode

Every GPIO can also be used in open-drain mode. To accomplish this, first, clear the respective bit in the `PORTx` or `PORTx_CLEAR` register then set the one bit in the `PORTx_INEN` register. Read from the `PORTx` register then return the status from the pin and do not return the state of the internal flip-flop.

By toggling the output driver through the `PORTx_DIR_SET` and `PORTx_DIR_CLEAR` register pair, the output signal can be pulled low or three-stated as required. Note that the polarity of the driven signal can be inverted when the internal flip-flop is set. When a GPIO port is used in open-drain mode, care must be taken not to exceed the  $V_{IH}$  operating condition associated with the respective pins.

## PORT Register List

Every port pin can operate in general-purpose I/O (GPIO) mode. This operation is the default after processor reset and is controlled by a set of registers that control GPIO functionality. Every bit in these registers represents a certain GPIO pin of a specific port. For more information on PORT functionality, see the PORT register descriptions.

**Table 14-1:** PORT Module Registers

| Name                       | Description                           |
|----------------------------|---------------------------------------|
| <code>PORT_FER</code>      | Port x Function Enable Register       |
| <code>PORT_FER_SET</code>  | Port x Function Enable Set Register   |
| <code>PORT_FER_CLR</code>  | Port x Function Enable Clear Register |
| <code>PORT_DATA</code>     | Port x GPIO Data Register             |
| <code>PORT_DATA_SET</code> | Port x GPIO Data Set Register         |
| <code>PORT_DATA_CLR</code> | Port x GPIO Data Clear Register       |

**Table 14-1:** PORT Module Registers (Continued)

| Name          | Description                                |
|---------------|--------------------------------------------|
| PORT_DIR      | Port x GPIO Direction Register             |
| PORT_DIR_SET  | Port x GPIO Direction Set Register         |
| PORT_DIR_CLR  | Port x GPIO Direction Clear Register       |
| PORT_INEN     | Port x GPIO Input Enable Register          |
| PORT_INEN_SET | Port x GPIO Input Enable Set Register      |
| PORT_INEN_CLR | Port x GPIO Input Enable Clear Register    |
| PORT_MUX      | Port x Multiplexer Control Register        |
| PORT_DATA_TGL | Port x GPIO Input Enable Toggle Register   |
| PORT_POL      | Port x GPIO Polarity Invert Register       |
| PORT_POL_SET  | Port x GPIO Polarity Invert Set Register   |
| PORT_POL_CLR  | Port x GPIO Polarity Invert Clear Register |
| PORT_LOCK     | Port x GPIO Lock Register                  |
| PORT_REVID    | Port x GPIO Revision ID                    |

## ADSP-BF60x PORT Muxing Table

**Table 14-2:** PORT Signal Muxing Table

| port A            |          |          |        |           |
|-------------------|----------|----------|--------|-----------|
| Pad ID / Mux Slot | 0        | 1        | 2      | Input Tap |
| PA_00 (A13)       | SMC0_A03 | PPI2_D00 | LP0_D0 |           |
| PA_01 (B13)       | SMC0_A04 | PPI2_D01 | LP0_D1 |           |
| PA_02 (A12)       | SMC0_A05 | PPI2_D02 | LP0_D2 |           |



**Table 14-2:** PORT Signal Muxing Table (Continued)

| port A            |          |          |        |           |
|-------------------|----------|----------|--------|-----------|
| Pad ID / Mux Slot | 0        | 1        | 2      | Input Tap |
| PA_03 (B12)       | SMC0_A06 | PPI2_D03 | LP0_D3 |           |
| PA_04 (A11)       | SMC0_A07 | PPI2_D04 | LP0_D4 |           |
| PA_05 (B11)       | SMC0_A08 | PPI2_D05 | LP0_D5 |           |
| PA_06 (A10)       | SMC0_A09 | PPI2_D06 | LP0_D6 |           |
| PA_07 (B10)       | SMC0_A10 | PPI2_D07 | LP0_D7 |           |
| PA_08 (A09)       | SMC0_A11 | PPI2_D08 | LP1_D0 |           |
| PA_09 (B09)       | SMC0_A12 | PPI2_D09 | LP1_D1 |           |
| PA_10 (A08)       | SMC0_A14 | PPI2_D10 | LP1_D2 |           |
| PA_11 (B08)       | SMC0_A15 | PPI2_D11 | LP1_D3 |           |
| PA_12 (A07)       | SMC0_A17 | PPI2_D12 | LP1_D4 |           |
| PA_13 (B07)       | SMC0_A18 | PPI2_D13 | LP1_D5 |           |
| PA_14 (A06)       | SMC0_A19 | PPI2_D14 | LP1_D6 |           |
| PA_15 (B06)       | SMC0_A20 | PPI2_D15 | LP1_D7 |           |

**Table 14-3:** PORT Signal Muxing Table

| port B            |             |          |         |           |
|-------------------|-------------|----------|---------|-----------|
| Pad ID / Mux Slot | 0           | 1        | 2       | Input Tap |
| PB_00 (C12)       | SMC0_NORCLK | PPI2_CLK | LP0_CLK |           |
| PB_01 (C11)       | SMC0_AMS1   | PPI2_FS1 | LP0_ACK |           |
| PB_02 (C10)       | SMC0_A13    | PPI2_FS2 | LP1_ACK |           |
| PB_03 (C09)       | SMC0_A16    | PPI2_FS3 | LP1_CLK |           |

**Table 14-3:** PORT Signal Muxing Table (Continued)

| port B            |             |            |           |           |
|-------------------|-------------|------------|-----------|-----------|
| Pad ID / Mux Slot | 0           | 1          | 2         | Input Tap |
| PB_04 (C08)       | SMC0_AMS2   | SMC0_ABE0  | SPT0_AFS  |           |
| PB_05 (C07)       | SMC0_AMS3   | SMC0_ABE1  | SPT0_ACLK |           |
| PB_06 (C06)       | SMC0_A21    | SPT0_ATDV  |           | TM0_ACLK4 |
| PB_07 (A05)       | SMC0_A22    | PPI2_D16   | SPT0_BFS  |           |
| PB_08 (B05)       | SMC0_A23    | PPI2_D17   | SPT0_BCLK |           |
| PB_09 (C05)       | SMC0_BGH    |            | SPT0_AD0  | TM0_ACLK2 |
| PB_10 (A04)       | SMC0_A24    |            | SPT0_BD1  | TM0_ACLK0 |
| PB_11 (B04)       | SMC0_A25    |            | SPT0_BD0  | TM0_ACLK3 |
| PB_12 (C04)       | SMC0_BG     | SPT0_BTDTV | SPT0_AD1  | TM0_ACLK1 |
| PB_13 (K03)       | ETH0_TXEN   | PPI1_FS1   |           | TM0_ACI6  |
| PB_14 (L03)       | ETH0_REFCLK | PPI1_CLK   |           |           |
| PB_15 (M03)       | ETH0_PTPPPS | PPI1_FS3   |           |           |

**Table 14-4:** PORT Signal Muxing Table

| port C            |            |          |   |           |
|-------------------|------------|----------|---|-----------|
| Pad ID / Mux Slot | 0          | 1        | 2 | Input Tap |
| PC_00 (K01)       | ETH0_RXD0  | PPI1_D00 |   |           |
| PC_01 (L02)       | ETH0_RXD1  | PPI1_D01 |   |           |
| PC_02 (L01)       | ETH0_TXD0  | PPI1_D02 |   |           |
| PC_03 (M02)       | ETH0_TXD1  | PPI1_D03 |   |           |
| PC_04 (M01)       | ETH0_RXERR | PPI1_D04 |   |           |

**Table 14-4:** PORT Signal Muxing Table (Continued)

| port C            |             |          |              |           |
|-------------------|-------------|----------|--------------|-----------|
| Pad ID / Mux Slot | 0           | 1        | 2            | Input Tap |
| PC_05 (N02)       | ETH0_CRS    | PPI1_D05 |              |           |
| PC_06 (N01)       | ETH0_MDC    | PPI1_D06 |              |           |
| PC_07 (P02)       | ETH0_MDIO   | PPI1_D07 |              |           |
| PC_08 (P01)       |             | PPI1_D08 |              |           |
| PC_09 (R02)       | ETH1_PTPPPS | PPI1_D09 |              |           |
| PC_10 (R01)       |             | PPI1_D10 |              |           |
| PC_11 (T02)       |             | PPI1_D11 | ETH_PTPAUXIN |           |
| PC_12 (T01)       | SPI0_SEL7   | PPI1_D12 |              |           |
| PC_13 (U02)       | SPI0_SEL6   | PPI1_D13 | ETH_PTPCLKIN |           |
| PC_14 (U01)       | SPI1_SEL7   | PPI1_D14 |              |           |
| PC_15 (V02)       | SPI0_SEL4   | PPI1_D15 |              |           |

**Table 14-5:** PORT Signal Muxing Table

| port D            |           |          |           |           |
|-------------------|-----------|----------|-----------|-----------|
| Pad ID / Mux Slot | 0         | 1        | 2         | Input Tap |
| PD_00 (V01)       | SPI0_D2   | PPI1_D16 | SPI0_SEL3 |           |
| PD_01 (W02)       | SPI0_D3   | PPI1_D17 | SPI0_SEL2 |           |
| PD_02 (Y02)       | SPI0_MISO |          |           |           |
| PD_03 (Y01)       | SPI0_MOSI |          |           |           |
| PD_04 (W01)       | SPI0_CLK  |          |           |           |
| PD_05 (AB02)      | SPI1_CLK  |          |           | TM0_ACLK7 |

**Table 14-5:** PORT Signal Muxing Table (Continued)

| port D            |             |           |           |           |
|-------------------|-------------|-----------|-----------|-----------|
| Pad ID / Mux Slot | 0           | 1         | 2         | Input Tap |
| PD_06 (P03)       | ETH0_PHYINT | PPI1_FS2  |           | TM0_ACI5  |
| PD_07 (R03)       |             | UART0_TX  |           | TM0_ACI3  |
| PD_08 (T03)       |             | UART0_RX  |           | TM0_ACI0  |
| PD_09 (U03)       | SPI0_SEL5   | UART0_RTS | SPI1_SEL4 |           |
| PD_10 (V03)       | SPI0_RDY    | UART0_CTS | SPI1_SEL3 |           |
| PD_11 (AA01)      | SPI0_SEL1   |           |           | SPI0_SS   |
| PD_12 (W03)       | SPI1_SEL1   | PPI0_D20  | SPT1_AD1  | SPI1_SS   |
| PD_13 (AA03)      | SPI1_MOSI   |           |           | TM0_ACLK5 |
| PD_14 (AB03)      | SPI1_MISO   |           |           | TM0_ACLK6 |
| PD_15 (Y04)       | SPI1_SEL2   | PPI0_D21  | SPT1_AD0  |           |

**Table 14-6:** PORT Signal Muxing Table

| port E            |          |          |                    |           |
|-------------------|----------|----------|--------------------|-----------|
| Pad ID / Mux Slot | 0        | 1        | 2                  | Input Tap |
| PE_00 (AA04)      | SPI1_D3  | PPI0_D18 | SPT1_BD1           |           |
| PE_01 (AB04)      | SPI1_D2  | PPI0_D19 | SPT1_BD0           |           |
| PE_02 (Y05)       | SPI1_RDY | PPI0_D22 | SPT1_ACLK          |           |
| PE_03 (AA05)      |          | PPI0_D16 | ACM0_FS/SPT1_BFS   |           |
| PE_04 (AB05)      |          | PPI0_D17 | ACM0_CLK/SPT1_BCLK |           |
| PE_05 (Y06)       |          | PPI0_D23 | SPT1_AFS           |           |

**Table 14-6:** PORT Signal Muxing Table (Continued)

| port E            |             |           |          |            |
|-------------------|-------------|-----------|----------|------------|
| Pad ID / Mux Slot | 0           | 1         | 2        | Input Tap  |
| PE_06 (Y07)       | SPT1_ATDV   | PPI0_FS3  | LP3_CLK  |            |
| PE_07 (Y08)       | SPT1_BTDTV  | PPI0_FS2  | LP3_ACK  |            |
| PE_08 (Y09)       | PWM0_SYNC   | PPI0_FS1  | LP2_ACK  | ACM0_T0    |
| PE_09 (Y10)       |             | PPI0_CLK  | LP2_CLK  | PWM0_TRIP0 |
| PE_10 (Y14)       | ETH1_MDC    | PWM1_DL   | RSI0_D6  |            |
| PE_11 (Y15)       | ETH1_MDIO   | PWM1_DH   | RSI0_D7  |            |
| PE_12 (Y13)       | ETH1_PHYINT | PWM1_CL   | RSI0_D5  |            |
| PE_13 (AB14)      | ETH1_CRS    | PWM1_CH   | RSI0_D4  |            |
| PE_14 (AA16)      | ETH1_RXERR  | SPT2_ATDV | TM0_TMR0 |            |
| PE_15 (AA15)      | ETH1_RXD1   | PWM1_BL   | RSI0_D3  |            |

**Table 14-7:** PORT Signal Muxing Table

| port F            |         |          |        |           |
|-------------------|---------|----------|--------|-----------|
| Pad ID / Mux Slot | 0       | 1        | 2      | Input Tap |
| PF_00 (AA13)      | PWM0_AL | PPI0_D00 | LP2_D0 |           |
| PF_01 (AB13)      | PWM0_AH | PPI0_D01 | LP2_D1 |           |
| PF_02 (AA12)      | PWM0_BL | PPI0_D02 | LP2_D2 |           |
| PF_03 (AB12)      | PWM0_BH | PPI0_D03 | LP2_D3 |           |
| PF_04 (AA11)      | PWM0_CL | PPI0_D04 | LP2_D4 |           |
| PF_05 (AB11)      | PWM0_CH | PPI0_D05 | LP2_D5 |           |
| PF_06 (AA10)      | PWM0_DL | PPI0_D06 | LP2_D6 |           |

**Table 14-7:** PORT Signal Muxing Table (Continued)

| port F            |           |          |        |            |
|-------------------|-----------|----------|--------|------------|
| Pad ID / Mux Slot | 0         | 1        | 2      | Input Tap  |
| PF_07 (AB10)      | PWM0_DH   | PPI0_D07 | LP2_D7 |            |
| PF_08 (AA09)      | SPI1_SEL5 | PPI0_D08 | LP3_D0 |            |
| PF_09 (AB09)      | SPI1_SEL6 | PPI0_D09 | LP3_D1 |            |
| PF_10 (AA08)      | ACM0_A4   | PPI0_D10 | LP3_D2 |            |
| PF_11 (AB08)      |           | PPI0_D11 | LP3_D3 | PWM0_TRIP1 |
| PF_12 (AA07)      | ACM0_A2   | PPI0_D12 | LP3_D4 |            |
| PF_13 (AB07)      | ACM0_A3   | PPI0_D13 | LP3_D5 |            |
| PF_14 (AA06)      | ACM0_A0   | PPI0_D14 | LP3_D6 |            |
| PF_15 (AB06)      | ACM0_A1   | PPI0_D15 | LP3_D7 |            |

**Table 14-8:** PORT Signal Muxing Table

| port G            |             |          |            |            |
|-------------------|-------------|----------|------------|------------|
| Pad ID / Mux Slot | 0           | 1        | 2          | Input Tap  |
| PG_00 (AA14)      | ETH1_RXD0   | PWM1_BH  | RSI0_D2    |            |
| PG_01 (Y17)       | SPT2_AFS    | TM0_TMR2 | CAN0_TX    |            |
| PG_02 (AB17)      | ETH1_TXD1   | PWM1_AL  | RSI0_D1    |            |
| PG_03 (AB15)      | ETH1_TXD0   | PWM1_AH  | RSI0_D0    |            |
| PG_04 (Y18)       | SPT2_ACLK   | TM0_TMR1 | CAN0_RX    | TM0_ACI2   |
| PG_05 (AA17)      | ETH1_TXEN   | RSI0_CMD | PWM1_SYNC  | ACM0_T1    |
| PG_06 (AB16)      | ETH1_REFCLK | RSI0_CLK | SPT2_BTDTV | PWM1_TRIP0 |
| PG_07 (AA19)      | SPT2_BFS    | TM0_TMR5 |            | CNT0_ZM    |

**Table 14-8:** PORT Signal Muxing Table (Continued)

| port G            |           |           |           |            |
|-------------------|-----------|-----------|-----------|------------|
| Pad ID / Mux Slot | 0         | 1         | 2         | Input Tap  |
| PG_08 (AA18)      | SPT2_AD1  | TM0_TMR3  |           | PWM1_TRIP1 |
| PG_09 (Y16)       | SPT2_AD0  | TM0_TMR4  |           |            |
| PG_10 (AB21)      | UART1_RTS | SPT2_BCLK |           |            |
| PG_11 (Y19)       | SPT2_BD1  | TM0_TMR6  |           | CNT0_UD    |
| PG_12 (AB18)      | SPT2_BD0  | TM0_TMR7  |           | CNT0_DG    |
| PG_13 (AA20)      | UART1_CTS |           |           | TM0_CLK    |
| PG_14 (AB19)      | UART1_RX  | SYS_IDLE1 |           | TM0_ACI1   |
| PG_15 (AB20)      | UART1_TX  | SYS_IDLE0 | SYS_SLEEP | TM0_ACI4   |

## PINT Register List

The pin-interrupt assignment (PINT) module controls the pin-to-interrupt assignment in a byte-wide manner. The pin-interrupt assignment registers do not consist of 32 individual bits. They consist of four control bytes, each functioning as a multiplexer control.

All PINT registers are 32 bits wide and can be accessed by 32-bit load/store instructions. They also support 16-bit operation where the upper 16 bits are ignored and the application uses the lower 16 bits only. Consequently, all PINT registers support 32-bit accesses as well as 16-bit accesses for the lower half words. Applications may use faster 16-bit accesses as long as they do not require functionality of upper register halves.

**Table 14-9:** PINT Module Registers

| Name          | Description              |
|---------------|--------------------------|
| PINT_MSK_SET  | Pint Mask Set Register   |
| PINT_MSK_CLR  | Pint Mask Clear Register |
| PINT_REQ      | Pint Request Register    |
| PINT_ASSIGN   | Pint Assign Register     |
| PINT_EDGE_SET | Pint Edge Set Register   |

**Table 14-9:** PINT Module Registers (Continued)

| Name          | Description                |
|---------------|----------------------------|
| PINT_EDGE_CLR | Pint Edge Clear Register   |
| PINT_INV_SET  | Pint Invert Set Register   |
| PINT_INV_CLR  | Pint Invert Clear Register |
| PINT_PINSTATE | Pint Pinstate Register     |
| PINT_LATCH    | Pint Latch Register        |

## ADSP-BF60x PINT Interrupt List

**Table 14-10:** ADSP-BF60x PINT Interrupt List

| Description               | ID | DMA Channel | Sensitivity |
|---------------------------|----|-------------|-------------|
| PINT0 Pin Interrupt Block | 21 |             | LEVEL       |
| PINT1 Pin Interrupt Block | 22 |             | LEVEL       |
| PINT2 Pin Interrupt Block | 23 |             | LEVEL       |
| PINT3 Pin Interrupt Block | 24 |             | LEVEL       |
| PINT4 Pin Interrupt Block | 25 |             | LEVEL       |
| PINT5 Pin Interrupt Block | 26 |             | LEVEL       |

## ADSP-BF60x PINT Trigger List

**Table 14-11:** Masters ADSP-BF60x PINT Trigger List

| Description               | ID | Sensitivity |
|---------------------------|----|-------------|
| PINT0 Pin Interrupt Block | 10 | LEVEL       |
| PINT1 Pin Interrupt Block | 11 | LEVEL       |
| PINT2 Pin Interrupt Block | 12 | LEVEL       |



**Table 14-11:** Masters ADSP-BF60x PINT Trigger List (Continued)

| Description               | ID | Sensitivity |
|---------------------------|----|-------------|
| PINT3 Pin Interrupt Block | 13 | LEVEL       |
| PINT4 Pin Interrupt Block | 14 | LEVEL       |
| PINT5 Pin Interrupt Block | 15 | LEVEL       |
|                           |    |             |

**Table 14-12:** Slaves ADSP-BF60x PINT Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

## PORT Definitions

This section provides definitions relating to the GPIO ports.

### *x* (*PORTx*)

The naming convention for bits uses a lowercase "x" to represent one of the existing ports alphabetically named beginning with A,B,C,... For example, the name `PORTx_REG` represents any one or all of `PORTA_REG`, `PORTB_REG`, `PORTC_REG`, and so on. The bit name `Px0` represents `PA0`, `PB0`, and so on.

## PORT Architectural Concepts

This section describes architectural concepts relating to the GPIO ports and signals, including external and internal interfaces.

## External Interfaces

The pin multiplexing hardware can be seen as a layer between the on-chip peripherals and the pads of the silicon. All port groups are controlled by this unit.

## Internal Interfaces

All MMR registers of the pin multiplexing, GPIO and pin interrupt control blocks can be accessed through the MMR bus. There is no DMA support. Every one of the pin interrupt modules has its own and dedicated interrupt request output signal that connects directly to the SIC controller.

## PORT Event Control

The following sections describe event generation in the PORT module.

## PORT Interrupt Signals

The pin interrupts are completely decoupled from GPIO functionality which has the following advantages.

- Flexible mapping scheme enables pins from up to four different ports to be grouped into one common interrupt scheme.
- Interrupts work on input and output pins regardless of whether in GPIO or functional mode.

The ADSP-BF60x Blackfin processors have a number of interrupt channels dedicated to pin interrupts. These channels are managed by a set of hardware blocks named PINT<sub>x</sub> (PINT0–PINT5). Every PINT<sub>x</sub> block can sense up to 32 GPIO pins as described in the following list and shown in the figure below.

- PINT0 can sense pins of PORTA and PORTB
- PINT1 can sense pins of PORTB and PORTC
- PINT2 can sense pins from PORTC and PORTD
- PINT3 can sense pins from PORTD and PORTE
- PINT4 can sense pins from PORTE and PORTF
- PINT5 can sense pins from PORTF and PORTG

Both 32-bit and 16-bit peripheral bus accesses to PINT<sub>x</sub> registers are supported.

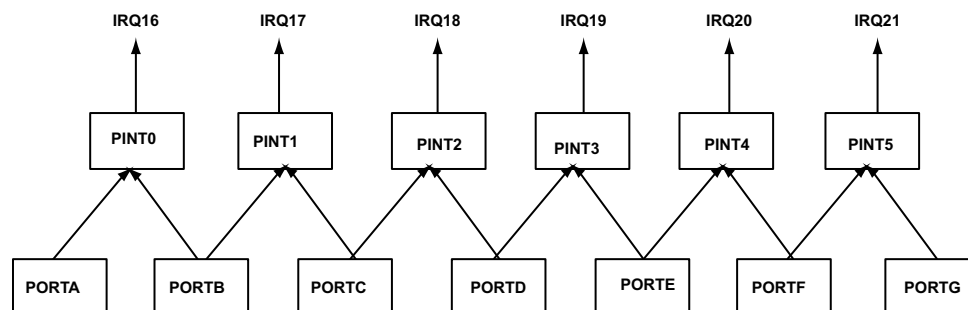


**Figure 14-2:** GPIO to PINT<sub>x</sub> Assignment

Pins are connected to the PINTx module and then to the system event controller. Special attention is required with regard to how the pins are assigned to the PINTx modules as shown in the PINTx block diagram below.

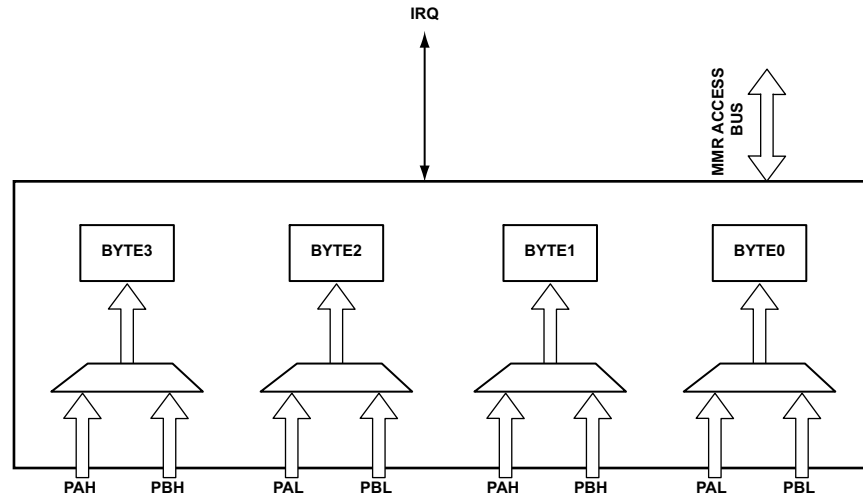


**Figure 14-3:** PINTx Block Diagram

The ports are subdivided into 8-bit half ports, with lower and upper half 8-bit units. The `PINTx_ASSIGN` registers control the 8-bit multiplexers shown in the Block Diagram. Lower half units of eight pins can be forwarded to either byte 0 or byte 2 of either associated PINTx block. Upper half units can be forwarded to either byte 1 or byte 3 of the pin interrupts blocks, without further restrictions.

When a half port is assigned to a byte in any PINTx block, the state of the eight pins (regardless of GPIO or function, input or output) can be seen in the `PINTx_PINSTATE` register. When neither input nor output drivers of the pin are enabled, the pin state is read as zero. The `PINTx_PINSTATE` register reports the inverted state of the pin if the signal inverter is activated by the `PINTx_INVERT_SET` register. The inverter can be enabled on a individual bit by bit basis. Every bit in the `PINTx_INVERT_SET/PINTx_INVERT_CLEAR` register pair represents a pin signal.

An interrupt can be generated on an active high level of the signal or a raising edge of the signal. The default behavior is level sensitivity. The `PINTx_EDGE_SET` register can be used to change the behavior to edge sensitivity. By enabling the inverter using the `PINTx_INVERT_SET` register, the interrupt behavior can be altered to trigger on active-low signals or falling edges.

The PINTx modules also assist if both signals are required to generate interrupts. If two different interrupt requests are required, the `PINTx_ASSIGN` registers can route a signal to two different PINTx blocks, where one block inverts the signal and the other one does not. If both signal edges can report over the same interrupt, every signal can be routed through to different bit positions within a single PINTx block, where the inverted signal should be enabled for either one. The servicing software routine can then tell from the `PINTx_LATCH` register whether a falling, a rising, or both edges have occurred.

Regardless of whether using level-sensitive or edge-sensitive mode, an interrupt is always latched by the hardware. Latched signals can be read from the `PINTx_LATCH` registers. Latches can only be cleared by a software or a hardware reset. To clear, W1C the `PINTx_REQUEST` or the `PINTx_LATCH` register. If the pin

state does not change by the time the interrupt service routine returns, the interrupt is requested again when in level-sensitive mode.

Because every PINTx block groups up to 32 pin signals, the PINTx\_MASK\_SET/PINTx\_INVERT\_CLEAR register pair can control which of the signals can request an interrupt at the system level. Software may interrogate the PINTx\_REQUEST register for signaling pins. The PINTx\_REQUEST bits represent a logical AND between the mask and the latch. When any of these bits is set, an interrupt is forwarded to the SIC controller.

All MMR registers in the pin interrupt module are 32 bits wide. Individual bits of the PINTx registers represent the associated pins. Nevertheless, the 32 bits can also be seen as four groups of eight pins. Each group can manage up to eight pins out of either the lower or an upper half of any associated port.

## PORT Programming Model

The following sections description of the overall program model of the general purpose ports.

*GPIO Programming Model Flow (Part 1)*, *GPIO Programming Model Flow (Part 2)*, and *GPIO Programming Model Flow (Part 3)* show the programming model of the general-purpose ports. This includes the GPIO input and output operation, open-drain mode, and the pin interrupt PINTx modules.

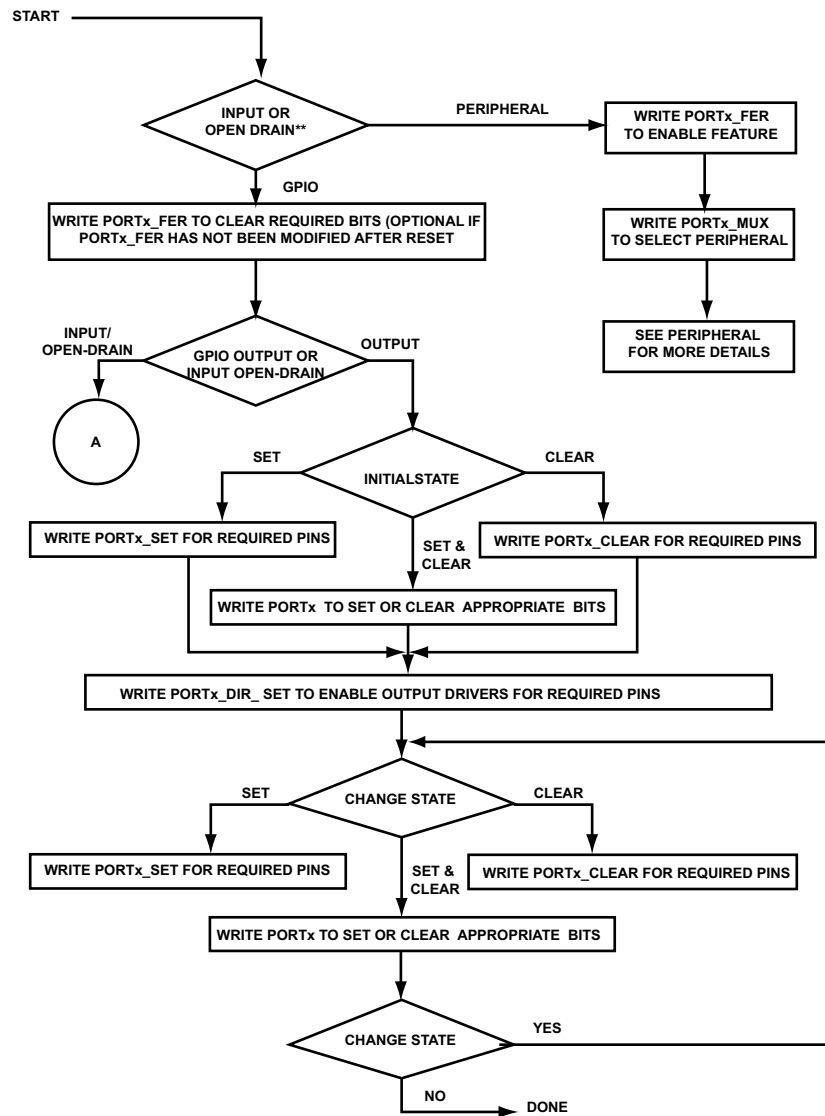
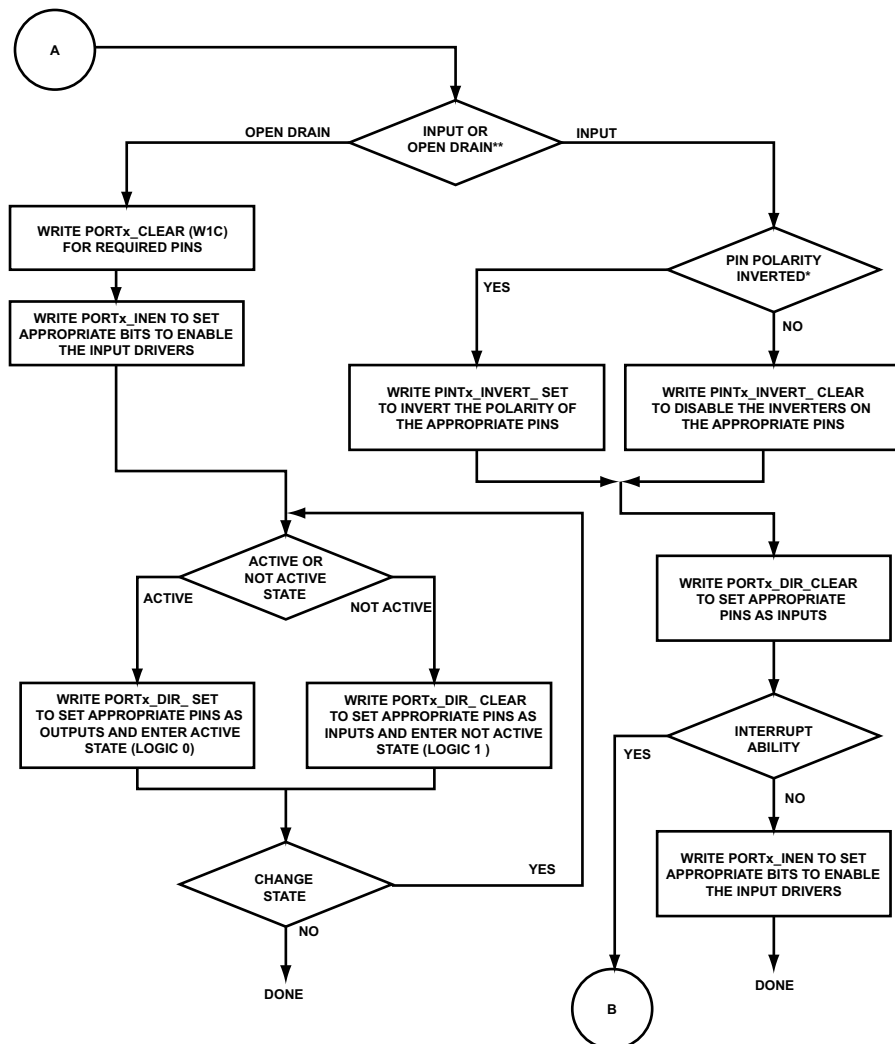


**Figure 14-4:** GPIO Programming Model Flow (Part 1)



\*The pin polarity set at this point will effect the behavior of the interrupt function described in the next figure. If the invert bit si set for a given pin, and edge sensitive interrupts are configured. The interrupt is latched on detection of a falling edge. If the inverse bit is clear, edge sensitive interrupts are generated on the rising edge. For level sensitive interrupts, enabling the inverter results in interrupts being detected on a low signal. Disabling the inverter results in interrupts being latched on high signals.

\*\*Open-drain mode assumes an external pull-up resistor is fitted.

**Figure 14-5:** GPIO Programming Model Flow (Part 2)

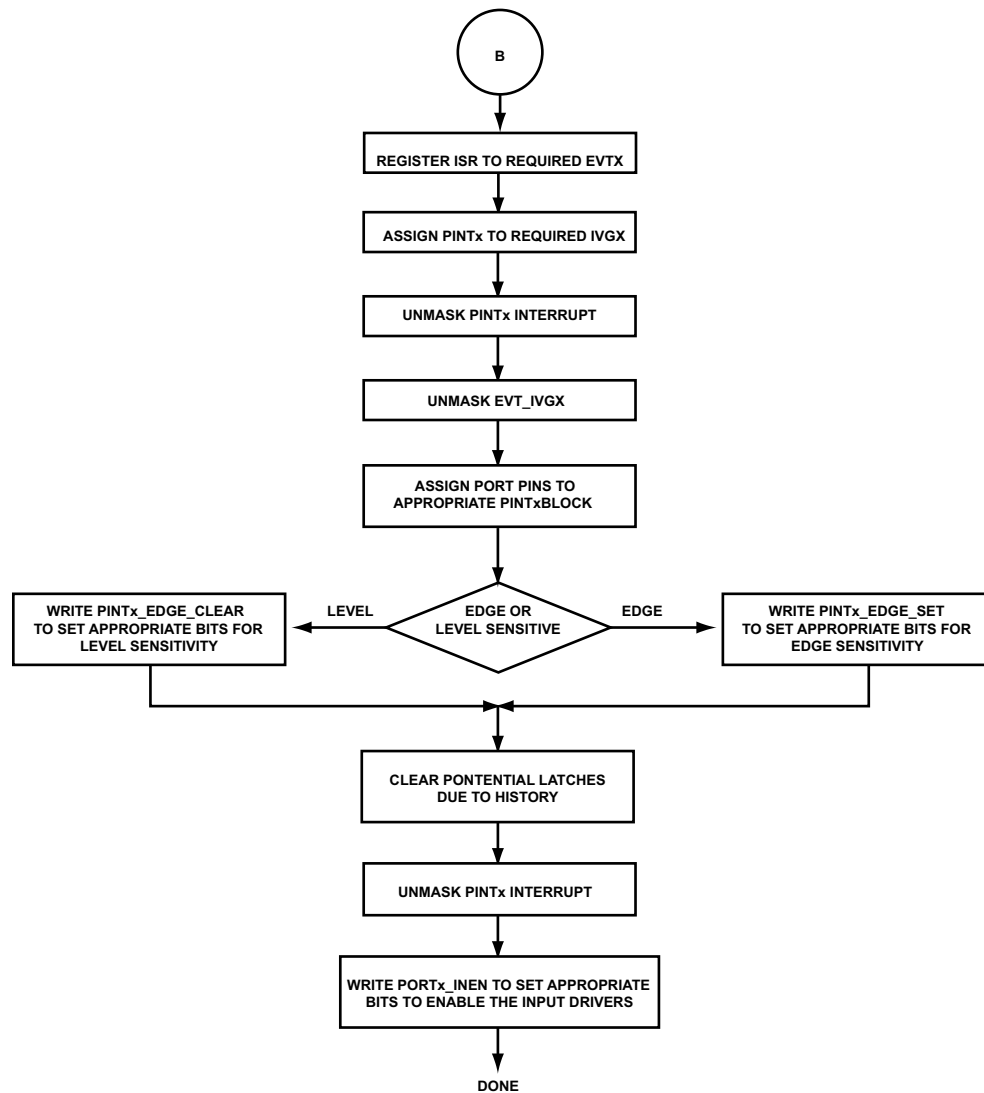


**Figure 14-6:** GPIO Programming Model Flow (Part 3)

## PORT Register Descriptions

General Purpose Input/Output (PORT) contains the following registers.

**Table 14-13:** PORT Module Registers

| Name         | Description                         |
|--------------|-------------------------------------|
| PORT_FER     | Port x Function Enable Register     |
| PORT_FER_SET | Port x Function Enable Set Register |

**Table 14-13:** PORT Module Registers (Continued)

| Name          | Description                                |
|---------------|--------------------------------------------|
| PORT_FER_CLR  | Port x Function Enable Clear Register      |
| PORT_DATA     | Port x GPIO Data Register                  |
| PORT_DATA_SET | Port x GPIO Data Set Register              |
| PORT_DATA_CLR | Port x GPIO Data Clear Register            |
| PORT_DIR      | Port x GPIO Direction Register             |
| PORT_DIR_SET  | Port x GPIO Direction Set Register         |
| PORT_DIR_CLR  | Port x GPIO Direction Clear Register       |
| PORT_INEN     | Port x GPIO Input Enable Register          |
| PORT_INEN_SET | Port x GPIO Input Enable Set Register      |
| PORT_INEN_CLR | Port x GPIO Input Enable Clear Register    |
| PORT_MUX      | Port x Multiplexer Control Register        |
| PORT_DATA_TGL | Port x GPIO Input Enable Toggle Register   |
| PORT_POL      | Port x GPIO Polarity Invert Register       |
| PORT_POL_SET  | Port x GPIO Polarity Invert Set Register   |
| PORT_POL_CLR  | Port x GPIO Polarity Invert Clear Register |
| PORT_LOCK     | Port x GPIO Lock Register                  |
| PORT_REVID    | Port x GPIO Revision ID                    |

## Port x Function Enable Register

The `PORT_FER` register bits indicate each port bit's operating mode: general purpose I/O mode or peripheral mode. After reset, all pins default to GPIO mode. Setting a bit in the `PORT_FER` registers enables a peripheral module to take ownership of the pin. The function enable bits impact output control only. Regardless of the setting of the function enable bits, both GPIO and peripherals can still sense the pin input. After a function is enabled, it is up to the `PORT_MUX` registers as to which peripheral takes control.



**PORT\_FER: Port x Function Enable Register - R/W**

Reset = 0x0000 0000

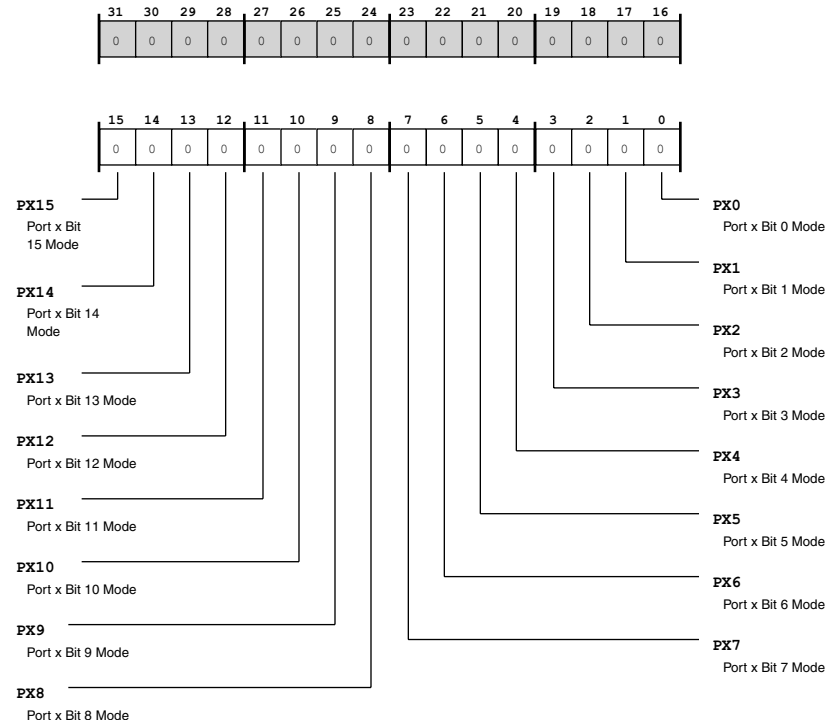


**Figure 14-7: PORT\_FER Register Diagram**

**Table 14-14: PORT\_FER Register Fields**

| Bit No. | Bit Name | Description/Enumeration |                 |
|---------|----------|-------------------------|-----------------|
| 15      | PX15     | Port x Bit 15 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 14      | PX14     | Port x Bit 14 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |

**Table 14-14:** PORT\_FER Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                 |
|---------|----------|-------------------------|-----------------|
| 13      | PX13     | Port x Bit 13 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 12      | PX12     | Port x Bit 12 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 11      | PX11     | Port x Bit 11 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 10      | PX10     | Port x Bit 10 Mode.     |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 9       | PX9      | Port x Bit 9 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 8       | PX8      | Port x Bit 8 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 7       | PX7      | Port x Bit 7 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |

**Table 14-14:** PORT\_FER Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                 |
|---------|----------|-------------------------|-----------------|
| 6       | PX6      | Port x Bit 6 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 5       | PX5      | Port x Bit 5 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 4       | PX4      | Port x Bit 4 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 3       | PX3      | Port x Bit 3 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 2       | PX2      | Port x Bit 2 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 1       | PX1      | Port x Bit 1 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |
| 0       | PX0      | Port x Bit 0 Mode.      |                 |
|         |          | 0                       | GPIO Mode       |
|         |          | 1                       | Peripheral Mode |

## Port x Function Enable Set Register

The PORT\_FER\_SET register permits enabling peripheral mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT\_FER\_SET enables peripheral mode for the corresponding pin.

PORT\_FER\_SET: Port x Function Enable Set Register - R/W

Reset = 0x0000 0000

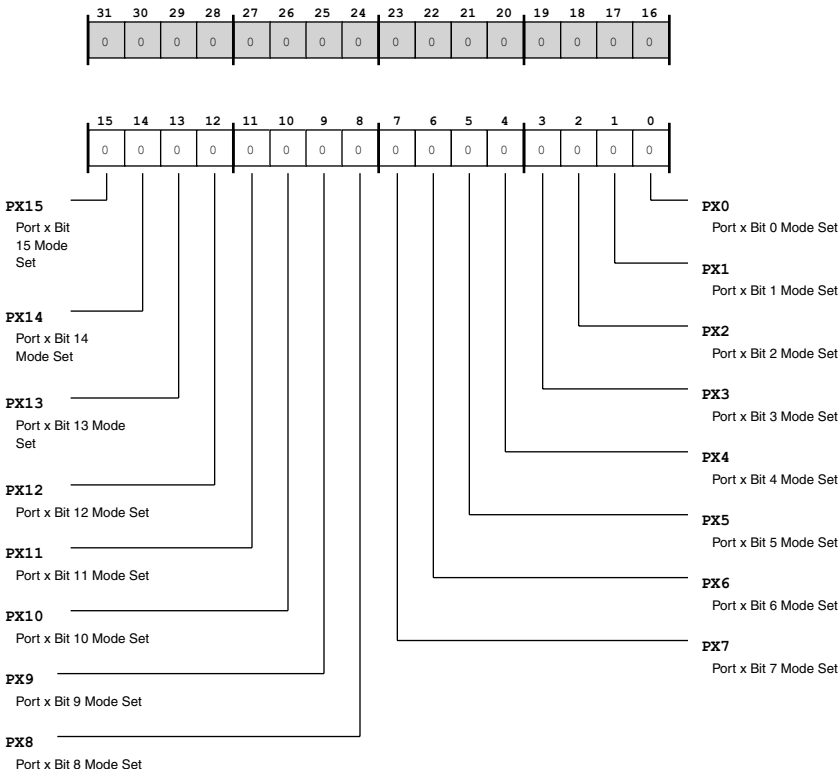


Figure 14-8: PORT\_FER\_SET Register Diagram

Table 14-15: PORT\_FER\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration |                             |
|---------|----------|-------------------------|-----------------------------|
| 15      | PX15     | Port x Bit 15 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |

**Table 14-15:** PORT\_FER\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                             |
|---------|----------|-------------------------|-----------------------------|
| 14      | PX14     | Port x Bit 14 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 13      | PX13     | Port x Bit 13 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 12      | PX12     | Port x Bit 12 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 11      | PX11     | Port x Bit 11 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 10      | PX10     | Port x Bit 10 Mode Set. |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 9       | PX9      | Port x Bit 9 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 8       | PX8      | Port x Bit 8 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |

**Table 14-15:** PORT\_FER\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                             |
|---------|----------|-------------------------|-----------------------------|
| 7       | PX7      | Port x Bit 7 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 6       | PX6      | Port x Bit 6 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 5       | PX5      | Port x Bit 5 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 4       | PX4      | Port x Bit 4 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 3       | PX3      | Port x Bit 3 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 2       | PX2      | Port x Bit 2 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |
| 1       | PX1      | Port x Bit 1 Mode Set.  |                             |
|         |          | 0                       | No Effect                   |
|         |          | 1                       | Set Bit for Peripheral Mode |

**Table 14-15:** PORT\_FER\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 0       | PX0      | Port x Bit 0 Mode Set.        |
|         |          | 0 No Effect                   |
|         |          | 1 Set Bit for Peripheral Mode |

## Port x Function Enable Clear Register

The PORT\_FER\_CLR register permits enabling GPIO mode for each bit and corresponding GPIO pin. Writing 1 to a bit in PORT\_FER\_CLR enables GPIO mode for the corresponding pin.

### PORT\_FER\_CLR: Port x Function Enable Clear Register - R/W

Reset = 0x0000 0000

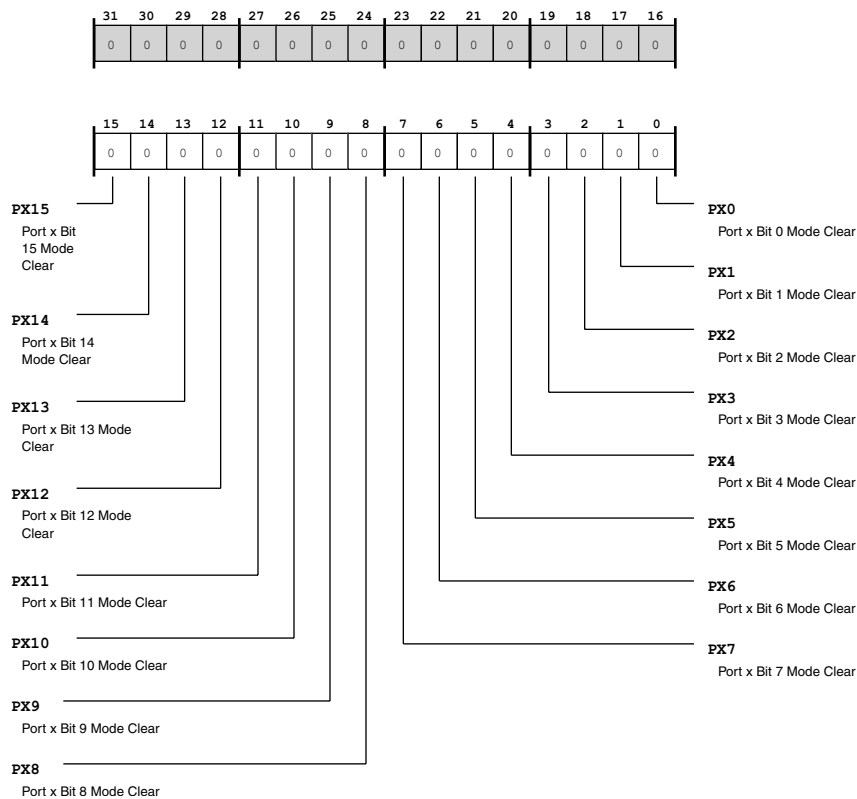


**Figure 14-9:** PORT\_FER\_CLR Register Diagram

**Table 14-16:** PORT\_FER\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration   |                       |
|---------|----------|---------------------------|-----------------------|
| 15      | PX15     | Port x Bit 15 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 14      | PX14     | Port x Bit 14 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 13      | PX13     | Port x Bit 13 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 12      | PX12     | Port x Bit 12 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 11      | PX11     | Port x Bit 11 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 10      | PX10     | Port x Bit 10 Mode Clear. |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |
| 9       | PX9      | Port x Bit 9 Mode Clear.  |                       |
|         |          | 0                         | No Effect             |
|         |          | 1                         | Set Bit for GPIO Mode |



**Table 14-16:** PORT\_FER\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration  |                       |
|---------|----------|--------------------------|-----------------------|
| 8       | PX8      | Port x Bit 8 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 7       | PX7      | Port x Bit 7 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 6       | PX6      | Port x Bit 6 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 5       | PX5      | Port x Bit 5 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 4       | PX4      | Port x Bit 4 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 3       | PX3      | Port x Bit 3 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 2       | PX2      | Port x Bit 2 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |

**Table 14-16:** PORT\_FER\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration  |                       |
|---------|----------|--------------------------|-----------------------|
| 1       | PX1      | Port x Bit 1 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |
| 0       | PX0      | Port x Bit 0 Mode Clear. |                       |
|         |          | 0                        | No Effect             |
|         |          | 1                        | Set Bit for GPIO Mode |

## Port x GPIO Data Register

The `PORT_DATA` register operates differently for port bits/pins, depending on whether the bit/pin is in output mode or input mode. In both modes, a set bit in the `PORT_DATA` register corresponds to a signal high on a GPIO pin, and a cleared bit in the `PORT_DATA` register corresponds to a signal low on a GPIO pin.

The `PORT_DATA`, `PORT_DATA_SET`, and `PORT_DATA_CLR` registers control the state of GPIO pins in output mode. To enable output mode (and output drivers), use the `PORT_DIR_SET` and `PORT_DIR_CLR` registers.

Writes to the `PORT_DATA` register affect the state of all pins of the port that are in output mode. To set or clear specific pins without impacting other pins of the port, use the `PORT_DATA_SET` and `PORT_DATA_CLR` registers.

When the GPIO pins are in input mode (input driver is enabled with the `PORT_INEN` register), reads from the `PORT_DATA`, `PORT_DATA_SET`, and `PORT_DATA_CLR` registers return the state of the respective GPIO pins.

Note that when the input driver is not enabled, reads from the `PORT_DATA`, `PORT_DATA_SET`, and `PORT_DATA_CLR` registers return the value previously written to the registers.

**PORT\_DATA: Port x GPIO Data Register - R/W**

Reset = 0x0000 0000

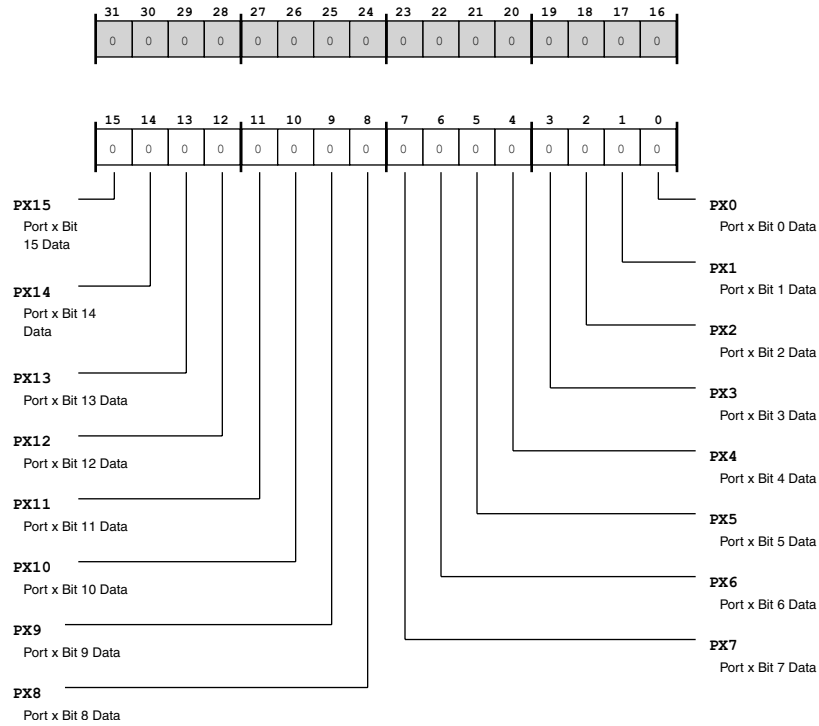


**Figure 14-10: PORT\_DATA Register Diagram**

**Table 14-17: PORT\_DATA Register Fields**

| Bit No. | Bit Name | Description/Enumeration |             |
|---------|----------|-------------------------|-------------|
| 15      | PX15     | Port x Bit 15 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 14      | PX14     | Port x Bit 14 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |

**Table 14-17:** PORT\_DATA Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |             |
|---------|----------|-------------------------|-------------|
| 13      | PX13     | Port x Bit 13 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 12      | PX12     | Port x Bit 12 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 11      | PX11     | Port x Bit 11 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 10      | PX10     | Port x Bit 10 Data.     |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 9       | PX9      | Port x Bit 9 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 8       | PX8      | Port x Bit 8 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 7       | PX7      | Port x Bit 7 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |

**Table 14-17:** PORT\_DATA Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |             |
|---------|----------|-------------------------|-------------|
| 6       | PX6      | Port x Bit 6 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 5       | PX5      | Port x Bit 5 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 4       | PX4      | Port x Bit 4 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 3       | PX3      | Port x Bit 3 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 2       | PX2      | Port x Bit 2 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 1       | PX1      | Port x Bit 1 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |
| 0       | PX0      | Port x Bit 0 Data.      |             |
|         |          | 0                       | Signal Low  |
|         |          | 1                       | Signal High |

## Port x GPIO Data Set Register

The PORT\_DATA\_SET register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT\_DATA register description.

PORT\_DATA\_SET: Port x GPIO Data Set Register - R/W

Reset = 0x0000 0000

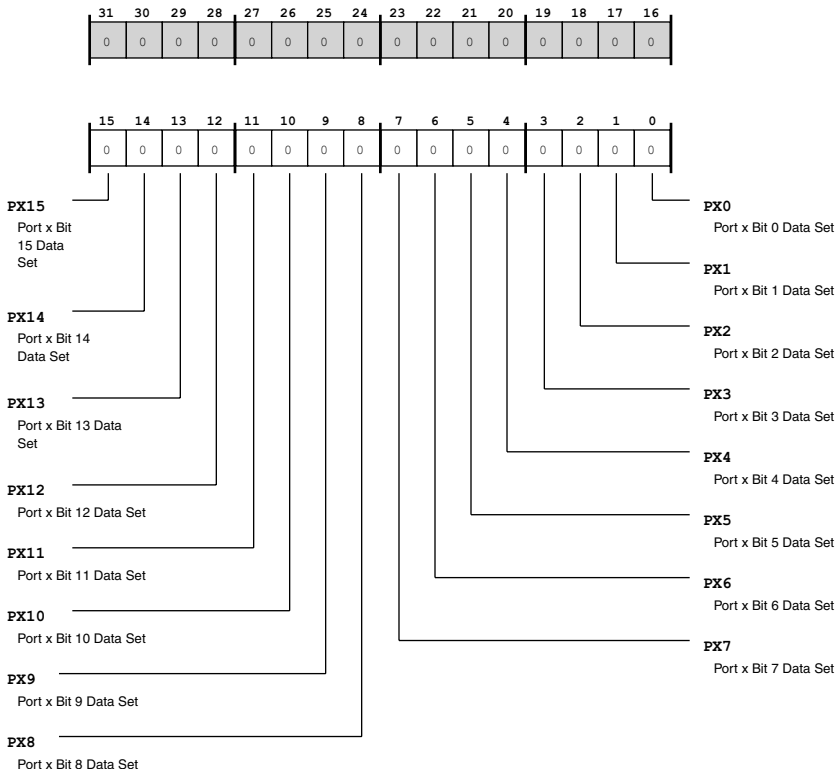


Figure 14-11: PORT\_DATA\_SET Register Diagram

Table 14-18: PORT\_DATA\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration |                                                    |
|---------|----------|-------------------------|----------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Data Set. |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |

**Table 14-18:** PORT\_DATA\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                                                               |
|---------|----------|-------------------------|-----------------------------------------------------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Data Set. |                                                                                               |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode.                                            |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode.                                            |
| 13      | PX13     | Port x Bit 13 Data Set. |                                                                                               |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode.                                            |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode.                                            |
| 12      | PX12     | Port x Bit 12 Data Set. |                                                                                               |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode.                                            |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode.<br>Write 1 for signal high in output mode. |
| 11      | PX11     | Port x Bit 11 Data Set. |                                                                                               |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode.                                            |
|         |          | 1                       | Set Bit                                                                                       |
| 10      | PX10     | Port x Bit 10 Data Set. |                                                                                               |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode.                                            |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode.                                            |

**Table 14-18:** PORT\_DATA\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                    |
|---------|----------|-------------------------|----------------------------------------------------|
| 9       | PX9      | Port x Bit 9 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 8       | PX8      | Port x Bit 8 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 7       | PX7      | Port x Bit 7 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 6       | PX6      | Port x Bit 6 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 5       | PX5      | Port x Bit 5 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |



**Table 14-18:** PORT\_DATA\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                    |
|---------|----------|-------------------------|----------------------------------------------------|
| 4       | PX4      | Port x Bit 4 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 3       | PX3      | Port x Bit 3 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 2       | PX2      | Port x Bit 2 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 1       | PX1      | Port x Bit 1 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |
| 0       | PX0      | Port x Bit 0 Data Set.  |                                                    |
|         |          | 0                       | No Effect<br>Write 0 has no effect in output mode. |
|         |          | 1                       | Set Bit<br>Write 1 for signal high in output mode. |

## Port x GPIO Data Clear Register

The PORT\_DATA\_CLR register operates differently for port bits/pins, depending on whether the bit/pin is output mode or input mode. For more information, see the PORT\_DATA register description.

PORT\_DATA\_CLR: Port x GPIO Data Clear Register - R/W

Reset = 0x0000 0000

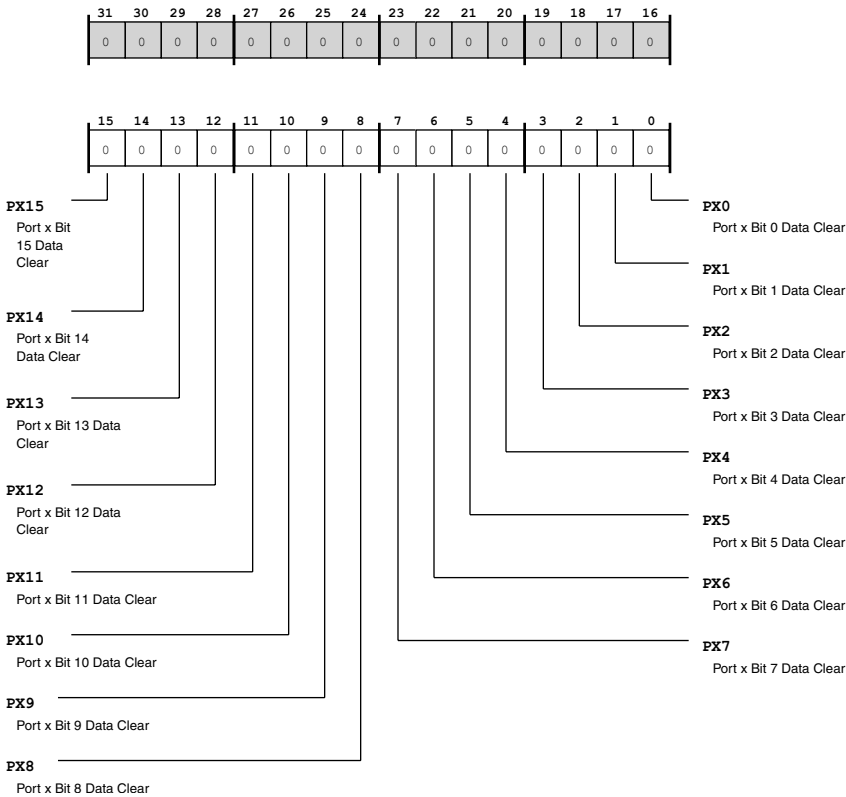


Figure 14-12: PORT\_DATA\_CLR Register Diagram

Table 14-19: PORT\_DATA\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration   |                                                     |
|---------|----------|---------------------------|-----------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Data Clear. |                                                     |
|         |          | 0                         | No Effect                                           |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode. |

**Table 14-19:** PORT\_DATA\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration   |                                                                                          |
|---------|----------|---------------------------|------------------------------------------------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Data Clear. |                                                                                          |
|         |          | 0                         | No Effect<br>Write 0 has no effect in output mode.                                       |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode.                                      |
| 13      | PX13     | Port x Bit 13 Data Clear. |                                                                                          |
|         |          | 0                         | No Effect<br>Write 0 has no effect in output mode.                                       |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode.                                      |
| 12      | PX12     | Port x Bit 12 Data Clear. |                                                                                          |
|         |          | 0                         | No Effect<br>Write 0 has no effect in output mode.                                       |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode.                                      |
| 11      | PX11     | Port x Bit 11 Data Clear. |                                                                                          |
|         |          | 0                         | No Effect                                                                                |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode.                                      |
| 10      | PX10     | Port x Bit 10 Data Clear. |                                                                                          |
|         |          | 0                         | No Effect<br>Write 0 has no effect in output mode. Write 0 has no effect in output mode. |
|         |          | 1                         | Clear Bit<br>Write 1 for signal low in output mode.                                      |

**Table 14-19:** PORT\_DATA\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration  |                                                     |
|---------|----------|--------------------------|-----------------------------------------------------|
| 9       | PX9      | Port x Bit 9 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 8       | PX8      | Port x Bit 8 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 7       | PX7      | Port x Bit 7 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 6       | PX6      | Port x Bit 6 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 5       | PX5      | Port x Bit 5 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |

**Table 14-19:** PORT\_DATA\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration  |                                                     |
|---------|----------|--------------------------|-----------------------------------------------------|
| 4       | PX4      | Port x Bit 4 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 3       | PX3      | Port x Bit 3 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 2       | PX2      | Port x Bit 2 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 1       | PX1      | Port x Bit 1 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |
| 0       | PX0      | Port x Bit 0 Data Clear. |                                                     |
|         |          | 0                        | No Effect<br>Write 0 has no effect in output mode.  |
|         |          | 1                        | Clear Bit<br>Write 1 for signal low in output mode. |

## Port x GPIO Direction Register

The `PORT_DIR`, `PORT_DIR_SET`, and `PORT_DIR_CLR` registers select output or input mode for GPIO pins and enable output drivers. Use the `PORT_INEN`, `PORT_INEN_SET`, and `PORT_INEN_CLR` registers to enable or disable input drivers.

Writes to the `PORT_DIR` register affect the state of all pins of the port. To select direction for specific pins without impacting other pins of the port, use the `PORT_DIR_SET` and `PORT_DIR_CLR` registers.

Setting a bit in the `PORT_DIR` register enables output mode on the corresponding a GPIO pin, and a clearing a bit in the `PORT_DIR` register disables output mode on the corresponding GPIO pin.

**Input Mode** - The default mode of every GPIO pin after reset is input mode, but the input drivers are not enabled. To enable any GPIO input drivers, set the corresponding bits in `PORT_INEN` register. When enabled, a read from the `PORT_DATA` register returns the logical state of the input pin. The input signal does not overwrite the state of the bit used for the output case. That state can only be altered by software. If the input driver is enabled, a write to the `PORT_DATA` register can alter the state of the bit, but the change cannot be read back.

**Output Mode** - Any GPIO pin can be configured for output mode. The GPIO output drivers are enabled by setting the corresponding bits in the `PORT_DIR`, `PORT_DIR_SET`, or `PORT_DIR_CLR` registers. By using the `PORT_DIR_SET` and `PORT_DIR_CLR` registers, direction of the signal flow of individual GPIO pins can be altered by separate software threads without mutually impacting other GPIOs on the same port. Both registers return the same value when read. Because the state of the GPIO output can already be controlled before the output driver is enabled, it is recommended to first set or clear the bit (using the `PORT_DATA`, `PORT_DATA_SET`, or `PORT_DATA_CLR` registers) to avoid any volatile levels on the output.

**Open-Drain Mode**- Every GPIO can also be used in open-drain mode. To accomplish this, first, clear the respective bit in the `PORT_DATA` or `PORT_DATA_CLR` register then set the one bit in the `PORT_INEN` register. Reads from the `PORT_DATA` register then return the status from the pin and do not return the state of the internal flip-flop. By toggling the output driver through the `PORT_DIR_SET` and `PORT_DIR_CLR` register pair, the output signal can be pulled low or three-stated as required. Note that the polarity of the driven signal can be inverted when the internal flip-flop is set instead. When a GPIO port is used in open-drain mode, care must be taken not to exceed the  $V_{IH}$  operating condition associated with the respective pin.

**PORT\_DIR: Port x GPIO Direction Register - R/W**

Reset = 0x0000 0000

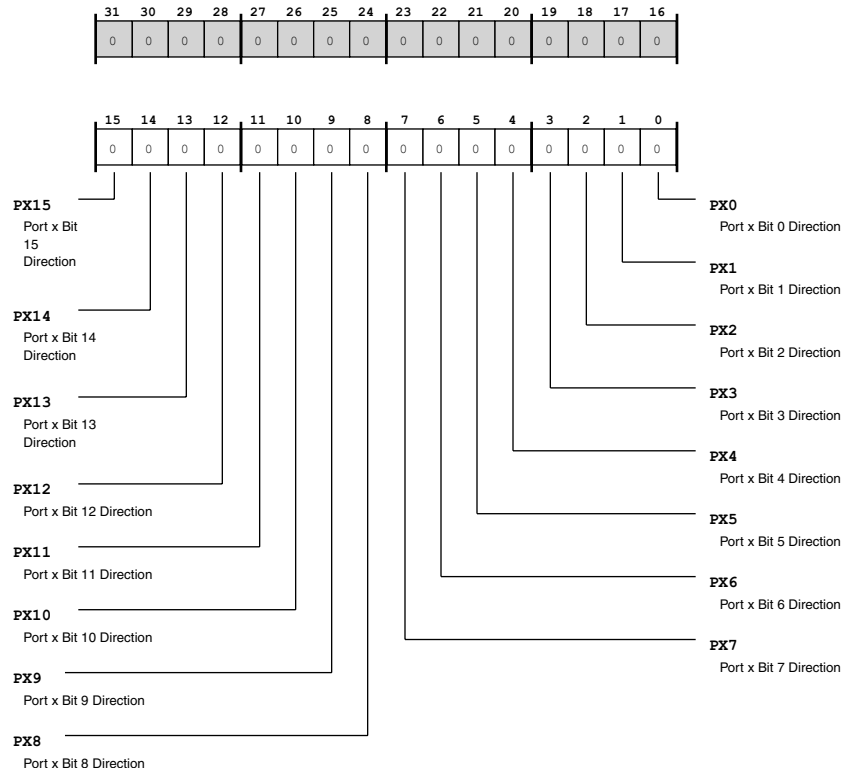


**Figure 14-13: PORT\_DIR Register Diagram**

**Table 14-20: PORT\_DIR Register Fields**

| Bit No. | Bit Name | Description/Enumeration  |                                       |
|---------|----------|--------------------------|---------------------------------------|
| 15      | PX15     | Port x Bit 15 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |
| 14      | PX14     | Port x Bit 14 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |

**Table 14-20:** PORT\_DIR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration  |                                       |
|---------|----------|--------------------------|---------------------------------------|
| 13      | PX13     | Port x Bit 13 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |
| 12      | PX12     | Port x Bit 12 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |
| 11      | PX11     | Port x Bit 11 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |
| 10      | PX10     | Port x Bit 10 Direction. |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |
| 9       | PX9      | Port x Bit 9 Direction.  |                                       |
|         |          | 0                        | Input mode<br>Output driver disabled. |
|         |          | 1                        | Output mode<br>Output driver enabled. |



**Table 14-20:** PORT\_DIR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                       |
|---------|----------|-------------------------|---------------------------------------|
| 8       | PX8      | Port x Bit 8 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 7       | PX7      | Port x Bit 7 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 6       | PX6      | Port x Bit 6 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 5       | PX5      | Port x Bit 5 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 4       | PX4      | Port x Bit 4 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |

**Table 14-20:** PORT\_DIR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                       |
|---------|----------|-------------------------|---------------------------------------|
| 3       | PX3      | Port x Bit 3 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 2       | PX2      | Port x Bit 2 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 1       | PX1      | Port x Bit 1 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |
| 0       | PX0      | Port x Bit 0 Direction. |                                       |
|         |          | 0                       | Input mode<br>Output driver disabled. |
|         |          | 1                       | Output mode<br>Output driver enabled. |

## Port x GPIO Direction Set Register

The `PORT_DIR_SET` register enable output mode and enables output drivers for GPIO pins. For more information, see the `PORT_DIR` register description.

**PORT\_DIR\_SET: Port x GPIO Direction Set Register - R/W**

Reset = 0x0000 0000

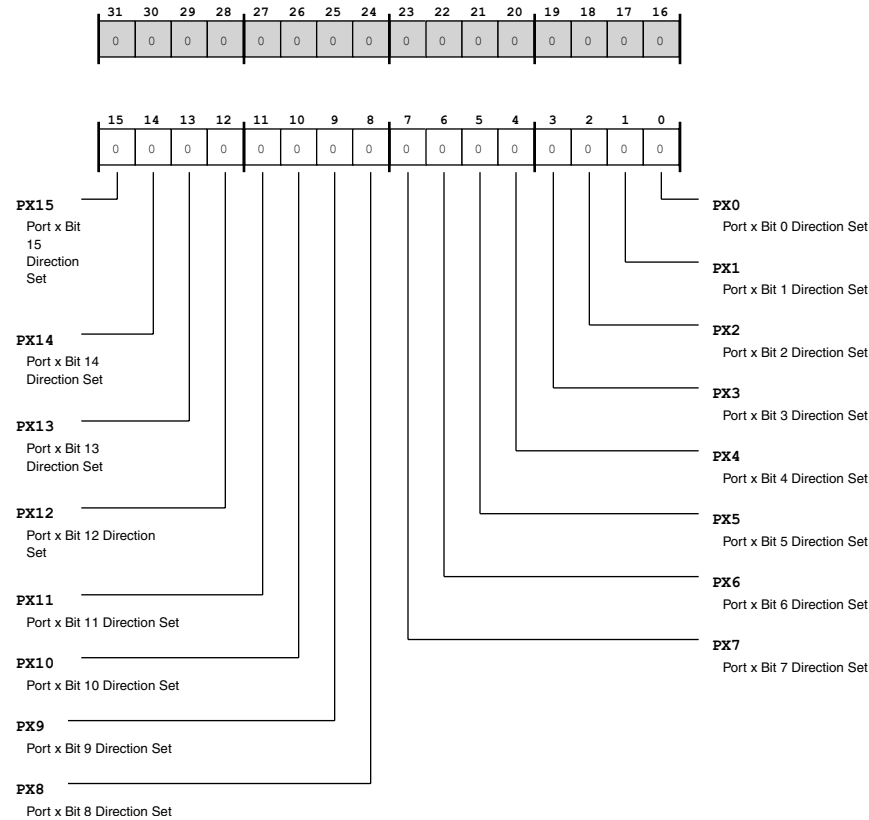


**Figure 14-14:** PORT\_DIR\_SET Register Diagram

**Table 14-21:** PORT\_DIR\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration      |                                              |
|---------|----------|------------------------------|----------------------------------------------|
| 15      | PX15     | Port x Bit 15 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 14      | PX14     | Port x Bit 14 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |

**Table 14-21:** PORT\_DIR\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration      |                                              |
|---------|----------|------------------------------|----------------------------------------------|
| 13      | PX13     | Port x Bit 13 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 12      | PX12     | Port x Bit 12 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 11      | PX11     | Port x Bit 11 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 10      | PX10     | Port x Bit 10 Direction Set. |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 9       | PX9      | Port x Bit 9 Direction Set.  |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |
| 8       | PX8      | Port x Bit 8 Direction Set.  |                                              |
|         |          | 0                            | No Effect                                    |
|         |          | 1                            | Set Bit<br>Set to enable output mode/driver. |

**Table 14-21:** PORT\_DIR\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration     |                                              |
|---------|----------|-----------------------------|----------------------------------------------|
| 7       | PX7      | Port x Bit 7 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 6       | PX6      | Port x Bit 6 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 5       | PX5      | Port x Bit 5 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 4       | PX4      | Port x Bit 4 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 3       | PX3      | Port x Bit 3 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 2       | PX2      | Port x Bit 2 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |

**Table 14-21:** PORT\_DIR\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration     |                                              |
|---------|----------|-----------------------------|----------------------------------------------|
| 1       | PX1      | Port x Bit 1 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |
| 0       | PX0      | Port x Bit 0 Direction Set. |                                              |
|         |          | 0                           | No Effect                                    |
|         |          | 1                           | Set Bit<br>Set to enable output mode/driver. |

## Port x GPIO Direction Clear Register

The PORT\_DIR\_CLR register disables output mode and disables output drivers for GPIO pins. For more information, see the PORT\_DIR register description.

PORT\_DIR\_CLR: Port x GPIO Direction Clear Register - R/W

Reset = 0x0000 0000

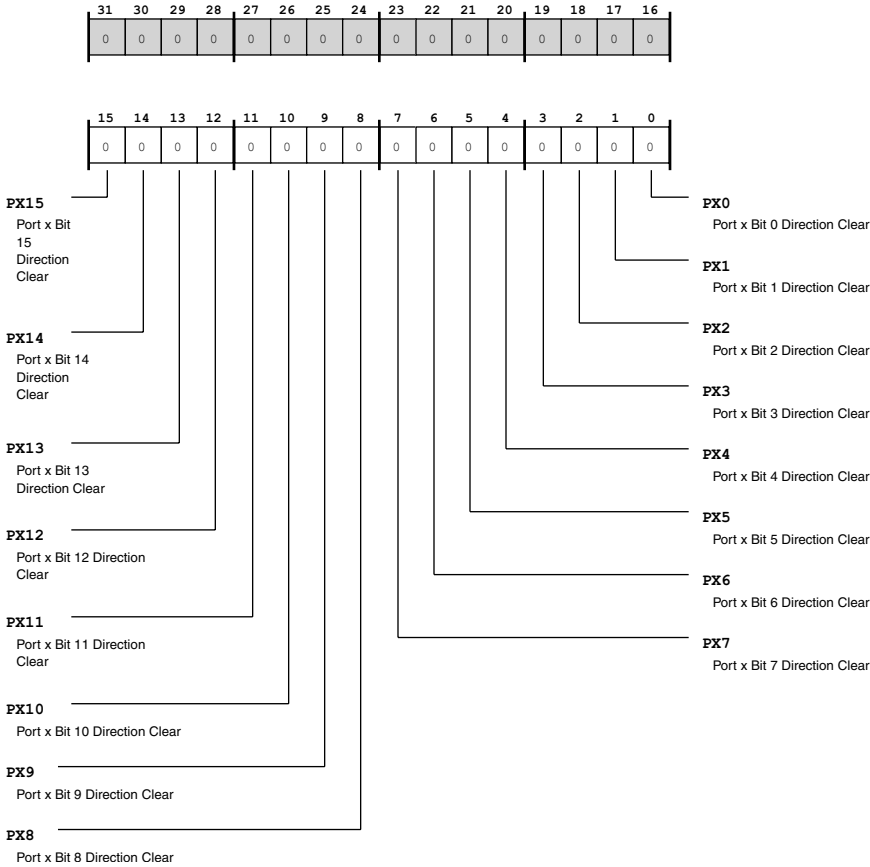


Figure 14-15: PORT\_DIR\_CLR Register Diagram

Table 14-22: PORT\_DIR\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration        |                                                 |
|---------|----------|--------------------------------|-------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |

**Table 14-22:** PORT\_DIR\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration        |                                                 |
|---------|----------|--------------------------------|-------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |
| 13      | PX13     | Port x Bit 13 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |
| 12      | PX12     | Port x Bit 12 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |
| 11      | PX11     | Port x Bit 11 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |
| 10      | PX10     | Port x Bit 10 Direction Clear. |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |
| 9       | PX9      | Port x Bit 9 Direction Clear.  |                                                 |
|         |          | 0                              | No Effect                                       |
|         |          | 1                              | Clear Bit<br>Set to disable output mode/driver. |



**Table 14-22:** PORT\_DIR\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration       |                                                 |
|---------|----------|-------------------------------|-------------------------------------------------|
| 8       | PX8      | Port x Bit 8 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 7       | PX7      | Port x Bit 7 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 6       | PX6      | Port x Bit 6 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 5       | PX5      | Port x Bit 5 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 4       | PX4      | Port x Bit 4 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 3       | PX3      | Port x Bit 3 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |

**Table 14-22:** PORT\_DIR\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration       |                                                 |
|---------|----------|-------------------------------|-------------------------------------------------|
| 2       | PX2      | Port x Bit 2 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 1       | PX1      | Port x Bit 1 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |
| 0       | PX0      | Port x Bit 0 Direction Clear. |                                                 |
|         |          | 0                             | No Effect                                       |
|         |          | 1                             | Clear Bit<br>Set to disable output mode/driver. |

## Port x GPIO Input Enable Register

The `PORT_INEN`, `PORT_INEN_SET`, and `PORT_INEN_CLR` registers enable or disable input drivers, which are required for using a GPIO pin in input mode.

Writes to the `PORT_INEN` register affect the input drivers for all pins of the port. To set or clear specific pin drivers without impacting other pin drivers of the port, use the `PORT_INEN_SET` and `PORT_INEN_CLR` registers.

If the input is enabled, reads from the `PORT_DATA`, `PORT_DATA_SET`, or `PORT_DATA_CLR` registers return the state of the pins. However, the state of the output is not overwritten by the input. It is altered by software writes only. Input and output drivers can be enabled at the same time. In this case, a read of the data register returns the true value of the data register and not the pin state.

For more information see the `PORT_DATA` register description and the `PORT_DIR` register description.

PORT\_INEN: Port x GPIO Input Enable Register - R/W

Reset = 0x0000 0000

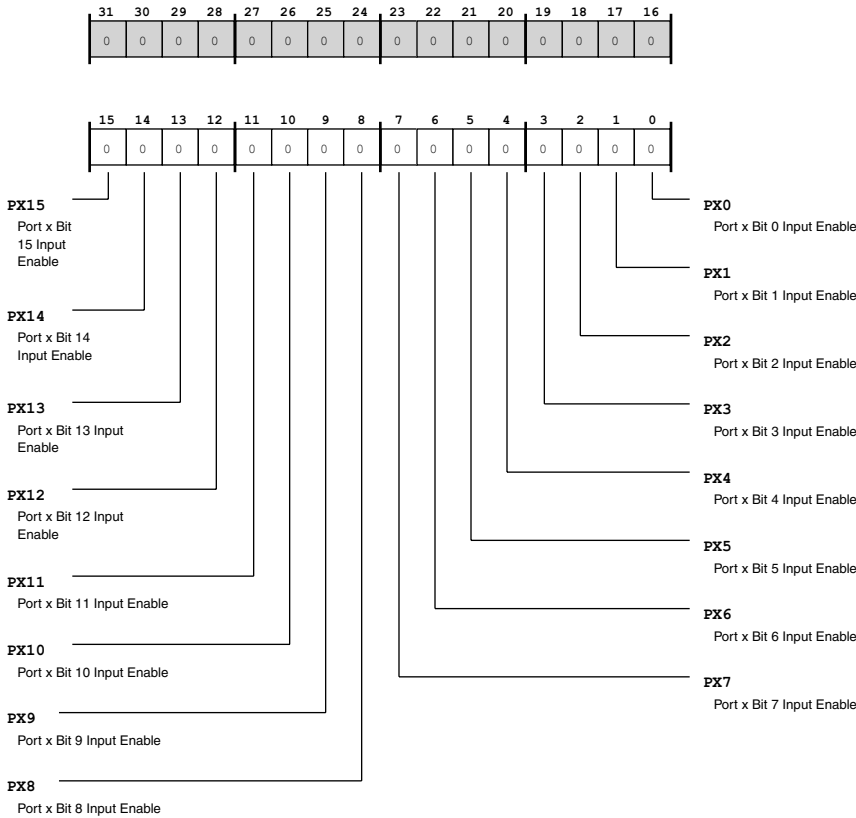


Figure 14-16: PORT\_INEN Register Diagram

Table 14-23: PORT\_INEN Register Fields

| Bit No. | Bit Name | Description/Enumeration     |                      |
|---------|----------|-----------------------------|----------------------|
| 15      | PX15     | Port x Bit 15 Input Enable. |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 14      | PX14     | Port x Bit 14 Input Enable. |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |

**Table 14-23:** PORT\_INEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration     |                      |
|---------|----------|-----------------------------|----------------------|
| 13      | PX13     | Port x Bit 13 Input Enable. |                      |
|         |          | 0                           | Input disabled       |
|         |          | 1                           | Enable Input Driver  |
| 12      | PX12     | Port x Bit 12 Input Enable. |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 11      | PX11     | Port x Bit 11 Input Enable. |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 10      | PX10     | Port x Bit 10 Input Enable. |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 9       | PX9      | Port x Bit 9 Input Enable.  |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 8       | PX8      | Port x Bit 8 Input Enable.  |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |
| 7       | PX7      | Port x Bit 7 Input Enable.  |                      |
|         |          | 0                           | Disable Input Driver |
|         |          | 1                           | Enable Input Driver  |

**Table 14-23:** PORT\_INEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration    |                      |
|---------|----------|----------------------------|----------------------|
| 6       | PX6      | Port x Bit 6 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 5       | PX5      | Port x Bit 5 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 4       | PX4      | Port x Bit 4 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 3       | PX3      | Port x Bit 3 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 2       | PX2      | Port x Bit 2 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 1       | PX1      | Port x Bit 1 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |
| 0       | PX0      | Port x Bit 0 Input Enable. |                      |
|         |          | 0                          | Disable Input Driver |
|         |          | 1                          | Enable Input Driver  |

## Port x GPIO Input Enable Set Register

The PORT\_INEN\_SET register enables input drivers for GPIO pins. For more information, see the PORT\_INEN register description.

**PORT\_INEN\_SET: Port x GPIO Input Enable Set Register - R/W**

Reset = 0x0000 0000

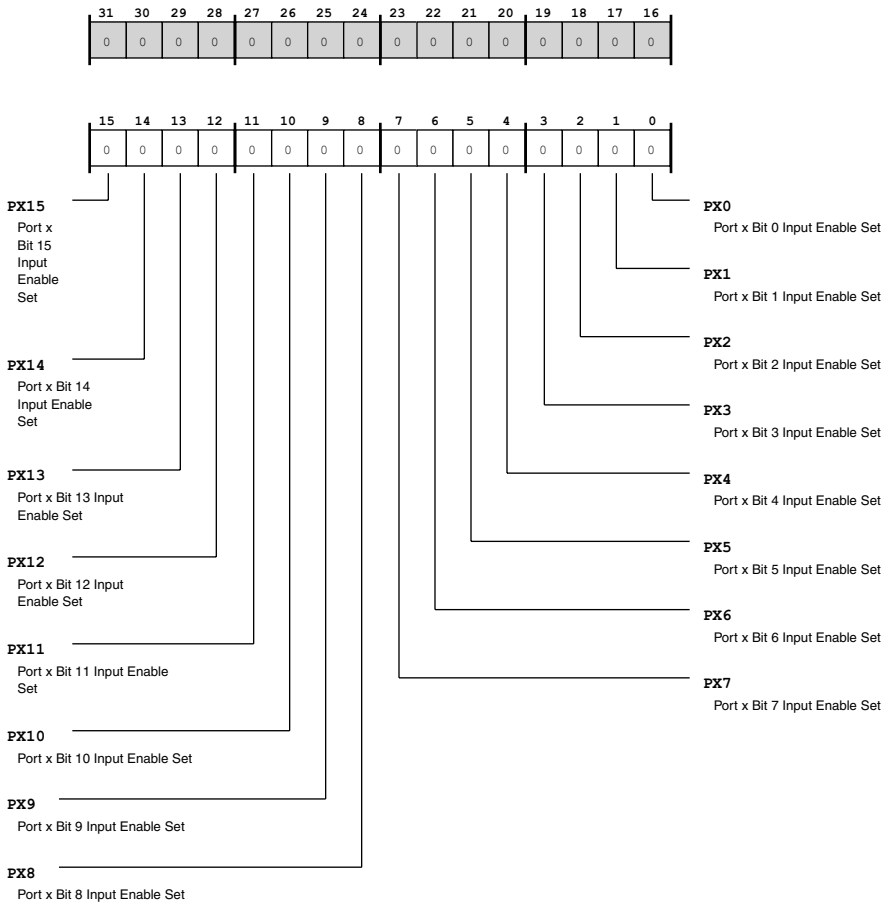


**Figure 14-17:** PORT\_INEN\_SET Register Diagram

**Table 14-24:** PORT\_INEN\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration         |                                        |
|---------|----------|---------------------------------|----------------------------------------|
| 15      | PX15     | Port x Bit 15 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |

**Table 14-24:** PORT\_INEN\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration         |                                        |
|---------|----------|---------------------------------|----------------------------------------|
| 14      | PX14     | Port x Bit 14 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |
| 13      | PX13     | Port x Bit 13 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |
| 12      | PX12     | Port x Bit 12 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |
| 11      | PX11     | Port x Bit 11 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |
| 10      | PX10     | Port x Bit 10 Input Enable Set. |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |
| 9       | PX9      | Port x Bit 9 Input Enable Set.  |                                        |
|         |          | 0                               | No Effect                              |
|         |          | 1                               | Set Bit<br>Set to enable input driver. |

**Table 14-24:** PORT\_INEN\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration        |                                        |
|---------|----------|--------------------------------|----------------------------------------|
| 8       | PX8      | Port x Bit 8 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 7       | PX7      | Port x Bit 7 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 6       | PX6      | Port x Bit 6 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 5       | PX5      | Port x Bit 5 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 4       | PX4      | Port x Bit 4 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 3       | PX3      | Port x Bit 3 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |



**Table 14-24:** PORT\_INEN\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration        |                                        |
|---------|----------|--------------------------------|----------------------------------------|
| 2       | PX2      | Port x Bit 2 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 1       | PX1      | Port x Bit 1 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |
| 0       | PX0      | Port x Bit 0 Input Enable Set. |                                        |
|         |          | 0                              | No Effect                              |
|         |          | 1                              | Set Bit<br>Set to enable input driver. |

## Port x GPIO Input Enable Clear Register

The PORT\_INEN\_CLR register disables input drivers for GPIO pins. For more information, see the PORT\_INEN register description.

PORT\_INEN\_CLR: Port x GPIO Input Enable Clear Register - R/W

Reset = 0x0000 0000

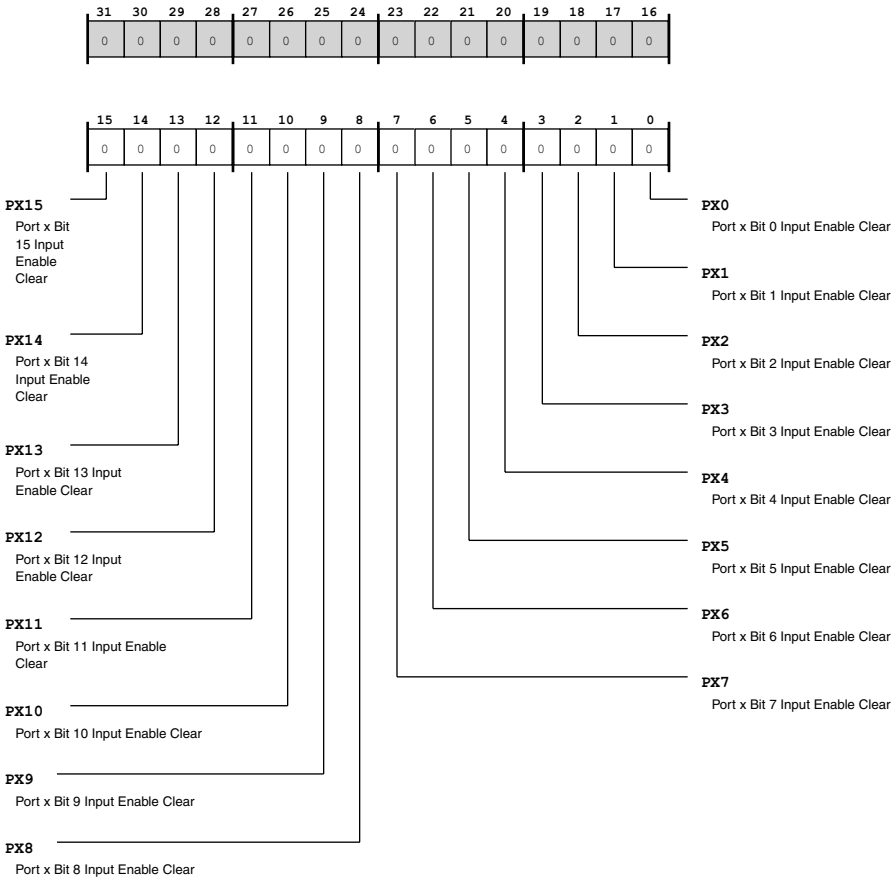


Figure 14-18: PORT\_INEN\_CLR Register Diagram

Table 14-25: PORT\_INEN\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration           |                                           |
|---------|----------|-----------------------------------|-------------------------------------------|
| 15      | PX15     | Port x Bit 15 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |

**Table 14-25:** PORT\_INEN\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration           |                                           |
|---------|----------|-----------------------------------|-------------------------------------------|
| 14      | PX14     | Port x Bit 14 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |
| 13      | PX13     | Port x Bit 13 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |
| 12      | PX12     | Port x Bit 12 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |
| 11      | PX11     | Port x Bit 11 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |
| 10      | PX10     | Port x Bit 10 Input Enable Clear. |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |
| 9       | PX9      | Port x Bit 9 Input Enable Clear.  |                                           |
|         |          | 0                                 | No Effect                                 |
|         |          | 1                                 | Clear Bit<br>Set to disable input driver. |

**Table 14-25:** PORT\_INEN\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration          |                                           |
|---------|----------|----------------------------------|-------------------------------------------|
| 8       | PX8      | Port x Bit 8 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 7       | PX7      | Port x Bit 7 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 6       | PX6      | Port x Bit 6 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 5       | PX5      | Port x Bit 5 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 4       | PX4      | Port x Bit 4 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 3       | PX3      | Port x Bit 3 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |

**Table 14-25:** PORT\_INEN\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration          |                                           |
|---------|----------|----------------------------------|-------------------------------------------|
| 2       | PX2      | Port x Bit 2 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 1       | PX1      | Port x Bit 1 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |
| 0       | PX0      | Port x Bit 0 Input Enable Clear. |                                           |
|         |          | 0                                | No Effect                                 |
|         |          | 1                                | Clear Bit<br>Set to disable input driver. |

## Port x Multiplexer Control Register

When a pin is in peripheral mode (not GPIO mode), the `PORT_MUX` register controls which peripheral takes ownership of a pin. Ports may have multiple, different peripheral functions. Two bits are required to describe every multiplexer on an individual pin-by-pin scheme. For example, Bit 0 and Bit 1 of the `PORT_MUX` register control the multiplexer of Pin 0, Bit 2 and Bit 3 of `PORT_MUX` control the multiplexer of Pin 1, and so on. The value of any `PORT_MUX` bit has no effect on the port pins when the associated bit in the `PORT_FER` register is 0 (selects GPIO mode). Even if a port has only one function, the `PORT_MUX` register is still present. For single function ports (no multiplexing is needed), leave the `PORT_MUX` bits at 0 (default). For all `PORT_MUX` bit fields: 00 = default/reset peripheral option, 01 = first alternate peripheral option, 10 = second alternate peripheral option, and 11 = third alternate peripheral option.

PORT\_MUX: Port x Multiplexer Control Register - R/W

Reset = 0x0000 0000

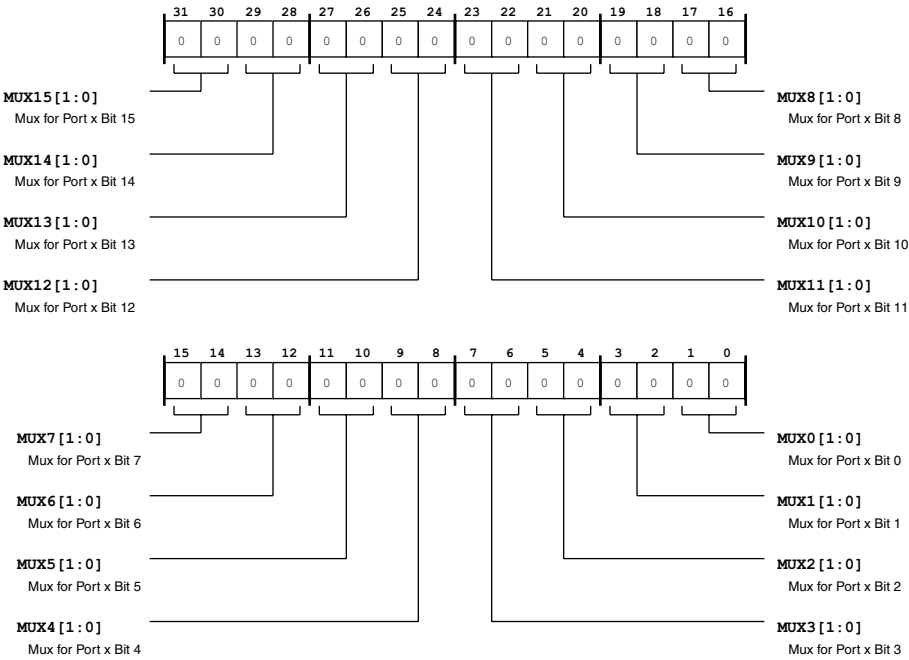


Figure 14-19: PORT\_MUX Register Diagram

Table 14-26: PORT\_MUX Register Fields

| Bit No. | Bit Name | Description/Enumeration                                          |
|---------|----------|------------------------------------------------------------------|
| 31:30   | MUX15    | Mux for Port x Bit 15.<br>Multiplexer control for Port x bit 15. |
| 29:28   | MUX14    | Mux for Port x Bit 14.<br>Multiplexer control for Port x bit 14. |
| 27:26   | MUX13    | Mux for Port x Bit 13.<br>Multiplexer control for Port x bit 13. |
| 25:24   | MUX12    | Mux for Port x Bit 12.<br>Multiplexer control for Port x bit 12. |
| 23:22   | MUX11    | Mux for Port x Bit 11.<br>Multiplexer control for Port x bit 11. |
| 21:20   | MUX10    | Mux for Port x Bit 10.<br>Multiplexer control for Port x bit 10. |

**Table 14-26:** PORT\_MUX Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                        |
|---------|----------|----------------------------------------------------------------|
| 19:18   | MUX9     | Mux for Port x Bit 9.<br>Multiplexer control for Port x bit 9. |
| 17:16   | MUX8     | Mux for Port x Bit 8.<br>Multiplexer control for Port x bit 8. |
| 15:14   | MUX7     | Mux for Port x Bit 7.<br>Multiplexer control for Port x bit 7. |
| 13:12   | MUX6     | Mux for Port x Bit 6.<br>Multiplexer control for Port x bit 6. |
| 11:10   | MUX5     | Mux for Port x Bit 5.<br>Multiplexer control for Port x bit 5. |
| 9:8     | MUX4     | Mux for Port x Bit 4.<br>Multiplexer control for Port x bit 4. |
| 7:6     | MUX3     | Mux for Port x Bit 3.<br>Multiplexer control for Port x bit 3. |
| 5:4     | MUX2     | Mux for Port x Bit 2.<br>Multiplexer control for Port x bit 2. |
| 3:2     | MUX1     | Mux for Port x Bit 1.<br>Multiplexer control for Port x bit 1. |
| 1:0     | MUX0     | Mux for Port x Bit 0.<br>Multiplexer control for Port x bit 0. |

## Port x GPIO Input Enable Toggle Register

The PORT\_DATA\_TGL register permits toggling the state of output GPIO pins. Setting bits in the PORT\_DATA\_TGL register affects the state of specific pins without impacting other pins of the port.

Reading the PORT\_DATA\_TGL returns the state of the PORT\_DATA register output pin state, but does not return the input pin/signal state.

PORT\_DATA\_TGL: Port x GPIO Input Enable Toggle Register - R/W

Reset = 0x0000 0000

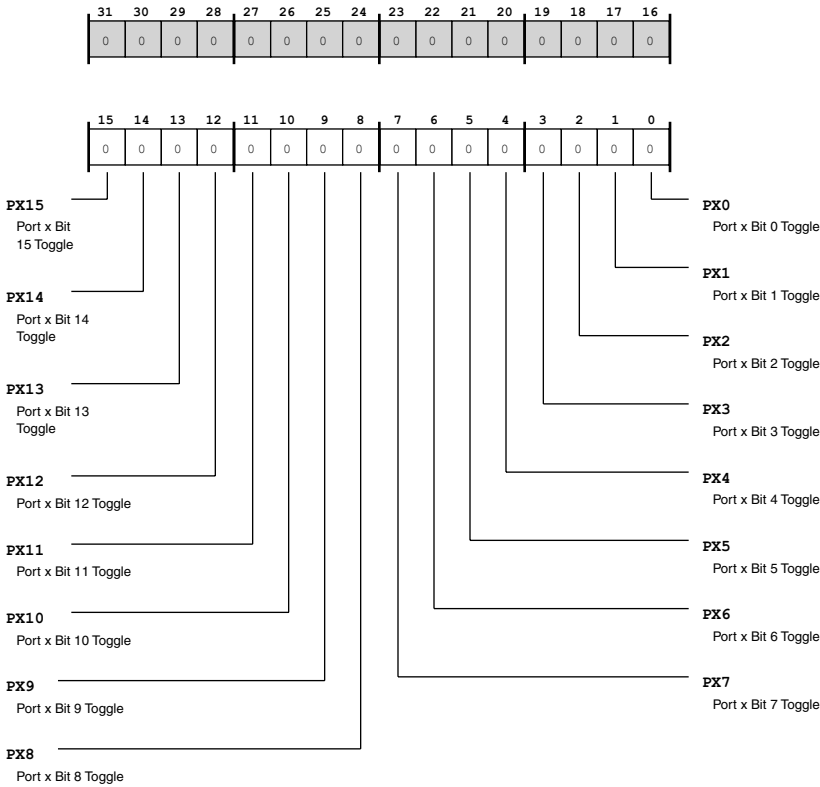


Figure 14-20: PORT\_DATA\_TGL Register Diagram

Table 14-27: PORT\_DATA\_TGL Register Fields

| Bit No. | Bit Name | Description/Enumeration |                                                        |
|---------|----------|-------------------------|--------------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 14      | PX14     | Port x Bit 14 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |



**Table 14-27:** PORT\_DATA\_TGL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                        |
|---------|----------|-------------------------|--------------------------------------------------------|
| 13      | PX13     | Port x Bit 13 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 12      | PX12     | Port x Bit 12 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 11      | PX11     | Port x Bit 11 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 10      | PX10     | Port x Bit 10 Toggle.   |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 9       | PX9      | Port x Bit 9 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 8       | PX8      | Port x Bit 8 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |

**Table 14-27:** PORT\_DATA\_TGL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                        |
|---------|----------|-------------------------|--------------------------------------------------------|
| 7       | PX7      | Port x Bit 7 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 6       | PX6      | Port x Bit 6 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 5       | PX5      | Port x Bit 5 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 4       | PX4      | Port x Bit 4 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 3       | PX3      | Port x Bit 3 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 2       | PX2      | Port x Bit 2 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |

**Table 14-27:** PORT\_DATA\_TGL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                                        |
|---------|----------|-------------------------|--------------------------------------------------------|
| 1       | PX1      | Port x Bit 1 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |
| 0       | PX0      | Port x Bit 0 Toggle.    |                                                        |
|         |          | 0                       | No Effect                                              |
|         |          | 1                       | Toggle Bit<br>Set to toggle output GPIO bit/pin state. |

## Port x GPIO Polarity Invert Register

The `PORT_POL`, `PORT_POL_SET`, and `PORT_POL_CLR` registers enable or disable inverting polarity of GPIO signals. To invert polarity of peripheral signals, use the inversion selection programming in the signal's corresponding module.

Writes to the `PORT_POL` register affect the polarity inversion selection of all pins of the port. To enable or disable polarity inversion for specific pins without impacting other pins of the port, use the `PORT_POL_SET` and `PORT_POL_CLR` registers.

Setting a bit in the `PORT_POL` register enables polarity inversion on the corresponding inversion GPIO pin, making the pin active-low or falling-edge sensitive. Clearing a bit in the `PORT_POL` register disables polarity (default state) on the corresponding GPIO pin, making it active-high or rising-edge sensitive.

PORT\_POL: Port x GPIO Polarity Invert Register - R/W

Reset = 0x0000 0000

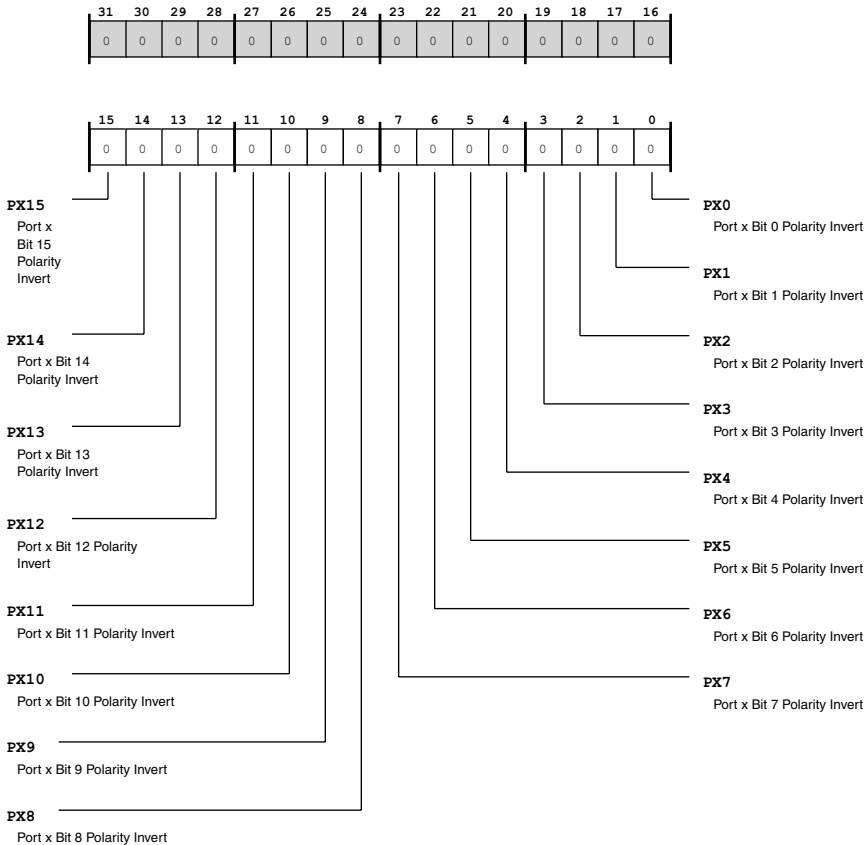


Figure 14-21: PORT\_POL Register Diagram

Table 14-28: PORT\_POL Register Fields

| Bit No. | Bit Name | Description/Enumeration        |                                                            |
|---------|----------|--------------------------------|------------------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |

**Table 14-28:** PORT\_POL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration        |                                                            |
|---------|----------|--------------------------------|------------------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |
| 13      | PX13     | Port x Bit 13 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |
| 12      | PX12     | Port x Bit 12 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |
| 11      | PX11     | Port x Bit 11 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |
| 10      | PX10     | Port x Bit 10 Polarity Invert. |                                                            |
|         |          | 0                              | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                              | Invert<br>GPIO is active low or falling edge sensitive.    |

**Table 14-28:** PORT\_POL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration       |                                                            |
|---------|----------|-------------------------------|------------------------------------------------------------|
| 9       | PX9      | Port x Bit 9 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 8       | PX8      | Port x Bit 8 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 7       | PX7      | Port x Bit 7 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 6       | PX6      | Port x Bit 6 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 5       | PX5      | Port x Bit 5 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |

**Table 14-28:** PORT\_POL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration       |                                                            |
|---------|----------|-------------------------------|------------------------------------------------------------|
| 4       | PX4      | Port x Bit 4 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 3       | PX3      | Port x Bit 3 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 2       | PX2      | Port x Bit 2 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 1       | PX1      | Port x Bit 1 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |
| 0       | PX0      | Port x Bit 0 Polarity Invert. |                                                            |
|         |          | 0                             | No Invert<br>GPIO is active high or rising edge sensitive. |
|         |          | 1                             | Invert<br>GPIO is active low or falling edge sensitive.    |

## Port x GPIO Polarity Invert Set Register

The PORT\_POL\_SET register enables polarity inversion for GPIO pins. For more information, see the PORT\_POL register description.

PORT\_POL\_SET: Port x GPIO Polarity Invert Set Register - R/W

Reset = 0x0000 0000

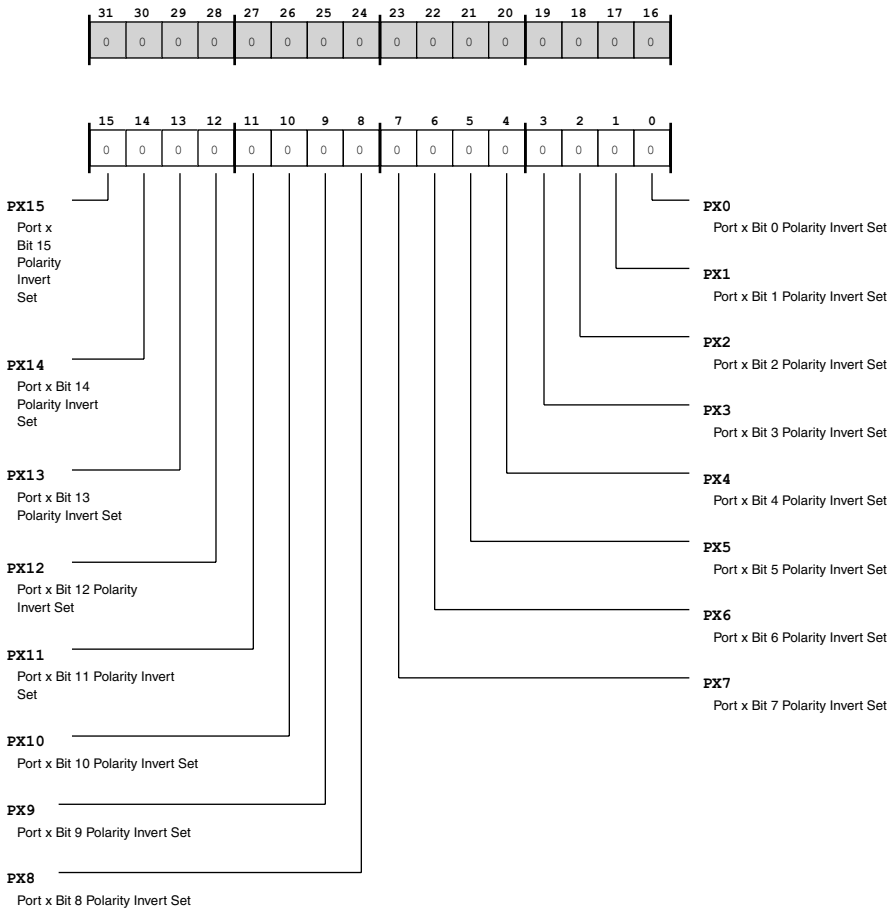


Figure 14-22: PORT\_POL\_SET Register Diagram

Table 14-29: PORT\_POL\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration            |                                                    |
|---------|----------|------------------------------------|----------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |



**Table 14-29:** PORT\_POL\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration            |                                                    |
|---------|----------|------------------------------------|----------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 13      | PX13     | Port x Bit 13 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 12      | PX12     | Port x Bit 12 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 11      | PX11     | Port x Bit 11 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 10      | PX10     | Port x Bit 10 Polarity Invert Set. |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 9       | PX9      | Port x Bit 9 Polarity Invert Set.  |                                                    |
|         |          | 0                                  | No Effect                                          |
|         |          | 1                                  | Set Bit<br>Set to enable GPIO pin polarity invert. |

**Table 14-29:** PORT\_POL\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration           |                                                    |
|---------|----------|-----------------------------------|----------------------------------------------------|
| 8       | PX8      | Port x Bit 8 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 7       | PX7      | Port x Bit 7 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 6       | PX6      | Port x Bit 6 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 5       | PX5      | Port x Bit 5 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 4       | PX4      | Port x Bit 4 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 3       | PX3      | Port x Bit 3 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |

**Table 14-29:** PORT\_POL\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration           |                                                    |
|---------|----------|-----------------------------------|----------------------------------------------------|
| 2       | PX2      | Port x Bit 2 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 1       | PX1      | Port x Bit 1 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |
| 0       | PX0      | Port x Bit 0 Polarity Invert Set. |                                                    |
|         |          | 0                                 | No Effect                                          |
|         |          | 1                                 | Set Bit<br>Set to enable GPIO pin polarity invert. |

## Port x GPIO Polarity Invert Clear Register

The PORT\_POL\_CLR register disables polarity inversion for GPIO pins. For more information, see the PORT\_POL register description.

PORT\_POL\_CLR: Port x GPIO Polarity Invert Clear Register - R/W

Reset = 0x0000 0000

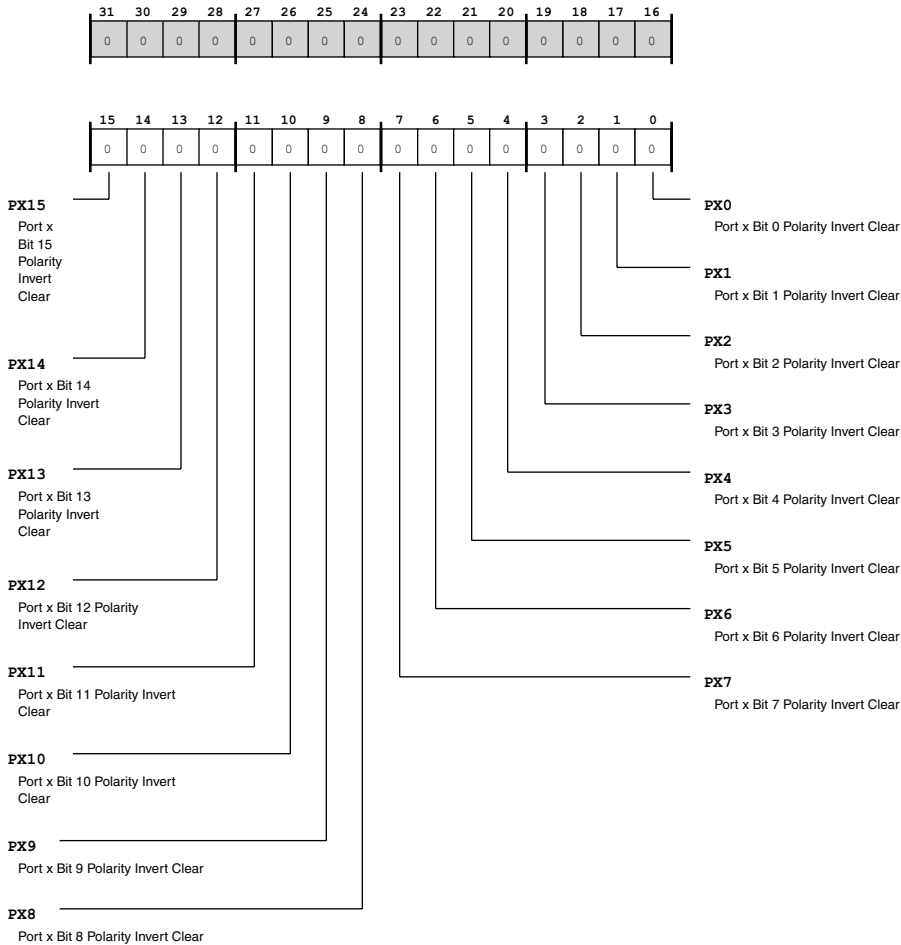


Figure 14-23: PORT\_POL\_CLR Register Diagram

Table 14-30: PORT\_POL\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration              |                                                       |
|---------|----------|--------------------------------------|-------------------------------------------------------|
| 15      | PX15     | Port x Bit 15 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |

**Table 14-30:** PORT\_POL\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration              |                                                       |
|---------|----------|--------------------------------------|-------------------------------------------------------|
| 14      | PX14     | Port x Bit 14 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 13      | PX13     | Port x Bit 13 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 12      | PX12     | Port x Bit 12 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 11      | PX11     | Port x Bit 11 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 10      | PX10     | Port x Bit 10 Polarity Invert Clear. |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 9       | PX9      | Port x Bit 9 Polarity Invert Clear.  |                                                       |
|         |          | 0                                    | No Effect                                             |
|         |          | 1                                    | Clear Bit<br>Set to disable GPIO pin polarity invert. |

**Table 14-30:** PORT\_POL\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration             |                                                       |
|---------|----------|-------------------------------------|-------------------------------------------------------|
| 8       | PX8      | Port x Bit 8 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 7       | PX7      | Port x Bit 7 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 6       | PX6      | Port x Bit 6 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 5       | PX5      | Port x Bit 5 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 4       | PX4      | Port x Bit 4 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 3       | PX3      | Port x Bit 3 Polarity Invert Clear. |                                                       |
|         |          | 0                                   | No Effect                                             |
|         |          | 1                                   | Clear Bit<br>Set to disable GPIO pin polarity invert. |

**Table 14-30:** PORT\_POL\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                 |
|---------|----------|---------------------------------------------------------|
| 2       | PX2      | Port x Bit 2 Polarity Invert Clear.                     |
|         |          | 0 No Effect                                             |
|         |          | 1 Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 1       | PX1      | Port x Bit 1 Polarity Invert Clear.                     |
|         |          | 0 No Effect                                             |
|         |          | 1 Clear Bit<br>Set to disable GPIO pin polarity invert. |
| 0       | PX0      | Port x Bit 0 Polarity Invert Clear.                     |
|         |          | 0 No Effect                                             |
|         |          | 1 Clear Bit<br>Set to disable GPIO pin polarity invert. |

## Port x GPIO Lock Register

The `PORT_LOCK` register enables (unlocks) or disables (locks) write access selectively for the PORT control registers.

### PORT\_LOCK: Port x GPIO Lock Register - R/W

Reset = 0x0000 0000

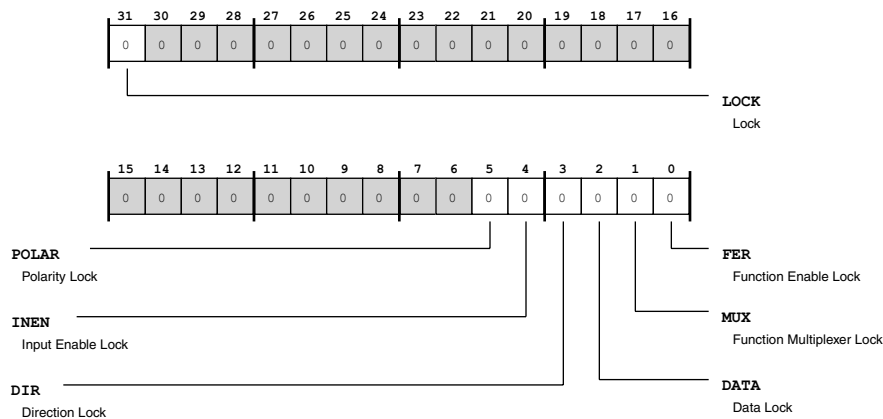


**Figure 14-24:** PORT\_LOCK Register Diagram

**Table 14-31:** PORT\_LOCK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the PORT_LOCK.LOCK bit is set, the PORT_LOCK register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                 |
|         |          | 1                      Lock                                                                                                                   |
| 5       | POLAR    | Polarity Lock.<br>The PORT_LOCK.POLAR disables write access to the PORT_POL, PORT_POL_SET, and PORT_POL_CLR registers.                        |
|         |          | 0                      Unlock POL                                                                                                             |
|         |          | 1                      Lock POL                                                                                                               |
| 4       | INEN     | Input Enable Lock.<br>The PORT_LOCK.INEN disables write access to the PORT_INEN, PORT_INEN_SET, and PORT_INEN_CLR registers.                  |
|         |          | 0                      Unlock INEN                                                                                                            |
|         |          | 1                      Lock INEN                                                                                                              |
| 3       | DIR      | Direction Lock.<br>The PORT_LOCK.DIR disables write access to the PORT_DIR, PORT_DIR_SET, PORT_DIR_CLR registers.                             |
|         |          | 0                      Lock DIR                                                                                                               |
|         |          | 1                      Unlock DIR                                                                                                             |
| 2       | DATA     | Data Lock.<br>The PORT_LOCK.DATA disables write access to the PORT_DATA, PORT_DATA_SET, PORT_DATA_CLR, and PORT_DATA_TGL registers.           |
|         |          | 0                      Unlock DATA                                                                                                            |
|         |          | 1                      Lock DATA                                                                                                              |



**Table 14-31:** PORT\_LOCK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------|
| 1       | MUX      | Function Multiplexer Lock.<br>The PORT_LOCK.MUX disables write accesses to the PORT_MUX register.                           |
|         |          | 0                      Unlock MUX                                                                                           |
|         |          | 1                      Lock MUX                                                                                             |
| 0       | FER      | Function Enable Lock.<br>The PORT_LOCK.FER disables write access to the PORT_FER, PORT_FER_SET, and PORT_FER_CLR registers. |
|         |          | 0                      Unlock FER                                                                                           |
|         |          | 1                      Lock FER                                                                                             |

Port x GPIO Revision ID

The PORT\_REVID register provides the PORT module revision information.

**PORT\_REVID: Port x GPIO Revision ID - R/WE**

Reset = 0x0000 0000

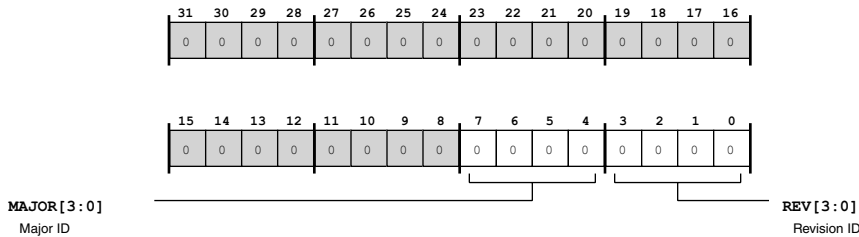


**Figure 14-25:** PORT\_REVID Register Diagram

**Table 14-32:** PORT\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:4     | MAJOR    | Major ID.               |
| 3:0     | REV      | Revision ID.            |

## PINT Register Descriptions

PINT (PINT) contains the following registers.

**Table 14-33:** PINT Module Registers

| Name          | Description                |
|---------------|----------------------------|
| PINT_MSK_SET  | Pint Mask Set Register     |
| PINT_MSK_CLR  | Pint Mask Clear Register   |
| PINT_REQ      | Pint Request Register      |
| PINT_ASSIGN   | Pint Assign Register       |
| PINT_EDGE_SET | Pint Edge Set Register     |
| PINT_EDGE_CLR | Pint Edge Clear Register   |
| PINT_INV_SET  | Pint Invert Set Register   |
| PINT_INV_CLR  | Pint Invert Clear Register |
| PINT_PINSTATE | Pint Pinstate Register     |
| PINT_LATCH    | Pint Latch Register        |

### Pint Mask Set Register

The PINT\_MSK\_SET register permits unmasking (enabling) of interrupts. Writing 1 to a bit in PINT\_MSK\_SET unmask the corresponding pin interrupt.

**PINT\_MSK\_SET: Pint Mask Set Register - R/W**

Reset = 0x0000 0000

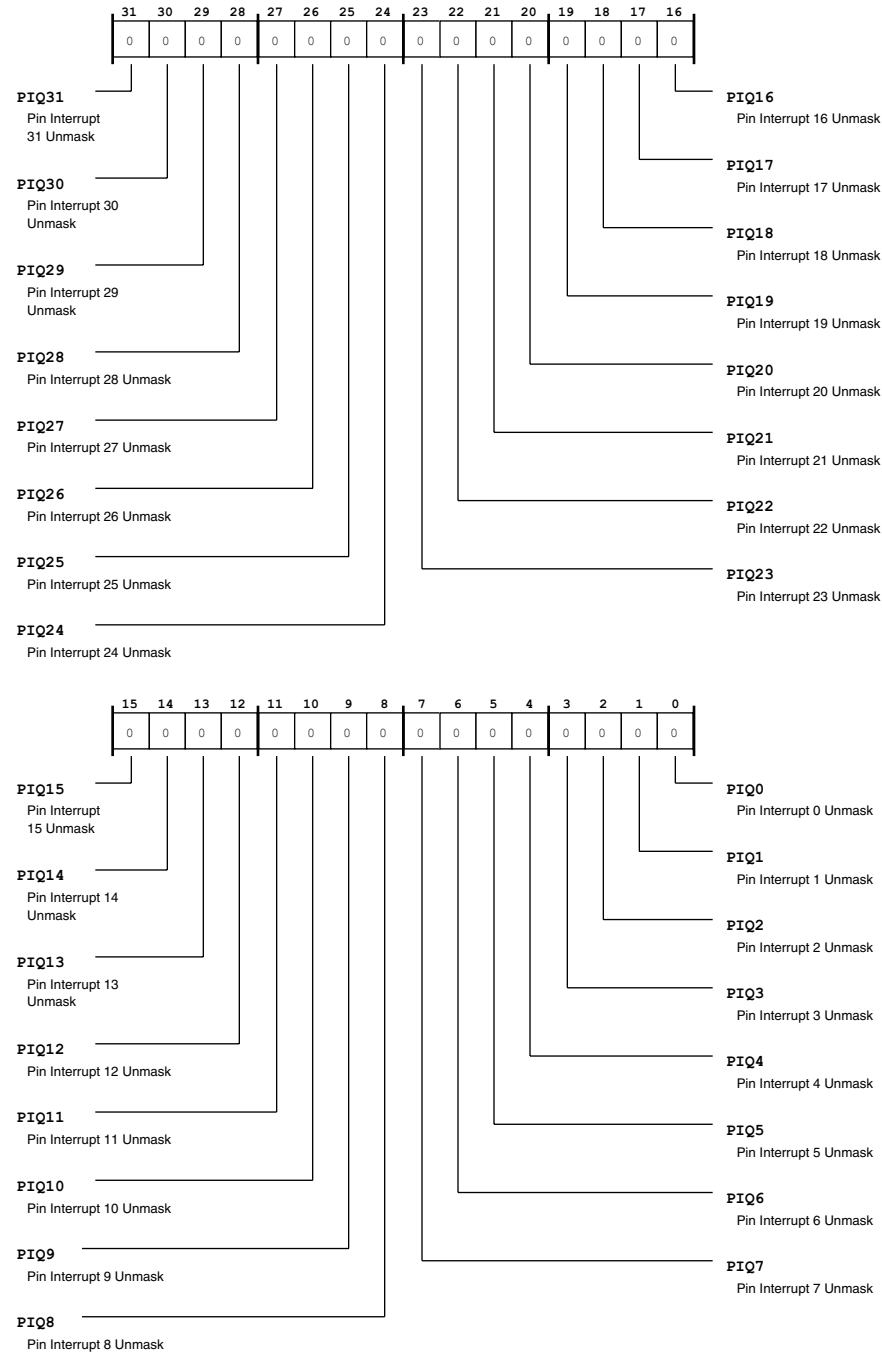


**Figure 14-26: PINT\_MSK\_SET Register Diagram**

**Table 14-34:** PINT\_MSK\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                              |
|---------|----------|------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Unmask.<br>Set to enable interrupt. |
| 30      | PIQ30    | Pin Interrupt 30 Unmask.<br>Set to enable interrupt. |
| 29      | PIQ29    | Pin Interrupt 29 Unmask.<br>Set to enable interrupt. |
| 28      | PIQ28    | Pin Interrupt 28 Unmask.<br>Set to enable interrupt. |
| 27      | PIQ27    | Pin Interrupt 27 Unmask.<br>Set to enable interrupt. |
| 26      | PIQ26    | Pin Interrupt 26 Unmask.<br>Set to enable interrupt. |
| 25      | PIQ25    | Pin Interrupt 25 Unmask.<br>Set to enable interrupt. |
| 24      | PIQ24    | Pin Interrupt 24 Unmask.<br>Set to enable interrupt. |
| 23      | PIQ23    | Pin Interrupt 23 Unmask.<br>Set to enable interrupt. |
| 22      | PIQ22    | Pin Interrupt 22 Unmask.<br>Set to enable interrupt. |
| 21      | PIQ21    | Pin Interrupt 21 Unmask.<br>Set to enable interrupt. |
| 20      | PIQ20    | Pin Interrupt 20 Unmask.<br>Set to enable interrupt. |
| 19      | PIQ19    | Pin Interrupt 19 Unmask.<br>Set to enable interrupt. |
| 18      | PIQ18    | Pin Interrupt 18 Unmask.<br>Set to enable interrupt. |

**Table 14-34:** PINT\_MSK\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                              |
|---------|----------|------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Unmask.<br>Set to enable interrupt. |
| 16      | PIQ16    | Pin Interrupt 16 Unmask.<br>Set to enable interrupt. |
| 15      | PIQ15    | Pin Interrupt 15 Unmask.<br>Set to enable interrupt. |
| 14      | PIQ14    | Pin Interrupt 14 Unmask.<br>Set to enable interrupt. |
| 13      | PIQ13    | Pin Interrupt 13 Unmask.<br>Set to enable interrupt. |
| 12      | PIQ12    | Pin Interrupt 12 Unmask.<br>Set to enable interrupt. |
| 11      | PIQ11    | Pin Interrupt 11 Unmask.<br>Set to enable interrupt. |
| 10      | PIQ10    | Pin Interrupt 10 Unmask.<br>Set to enable interrupt. |
| 9       | PIQ9     | Pin Interrupt 9 Unmask.<br>Set to enable interrupt.  |
| 8       | PIQ8     | Pin Interrupt 8 Unmask.<br>Set to enable interrupt.  |
| 7       | PIQ7     | Pin Interrupt 7 Unmask.<br>Set to enable interrupt.  |
| 6       | PIQ6     | Pin Interrupt 6 Unmask.<br>Set to enable interrupt.  |
| 5       | PIQ5     | Pin Interrupt 5 Unmask.<br>Set to enable interrupt.  |
| 4       | PIQ4     | Pin Interrupt 4 Unmask.<br>Set to enable interrupt.  |

**Table 14-34:** PINT\_MSK\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Unmask.<br>Set to enable interrupt. |
| 2       | PIQ2     | Pin Interrupt 2 Unmask.<br>Set to enable interrupt. |
| 1       | PIQ1     | Pin Interrupt 1 Unmask.<br>Set to enable interrupt. |
| 0       | PIQ0     | Pin Interrupt 0 Unmask.<br>Set to enable interrupt. |

## Pint Mask Clear Register

The PINT\_MSK\_CLR register permits masking (disabling) of interrupts. Writing 1 to a bit in PINT\_MSK\_CLR masks the corresponding pin interrupt.

**PINT\_MSK\_CLR: Pint Mask Clear Register - R/W**

Reset = 0x0000 0000

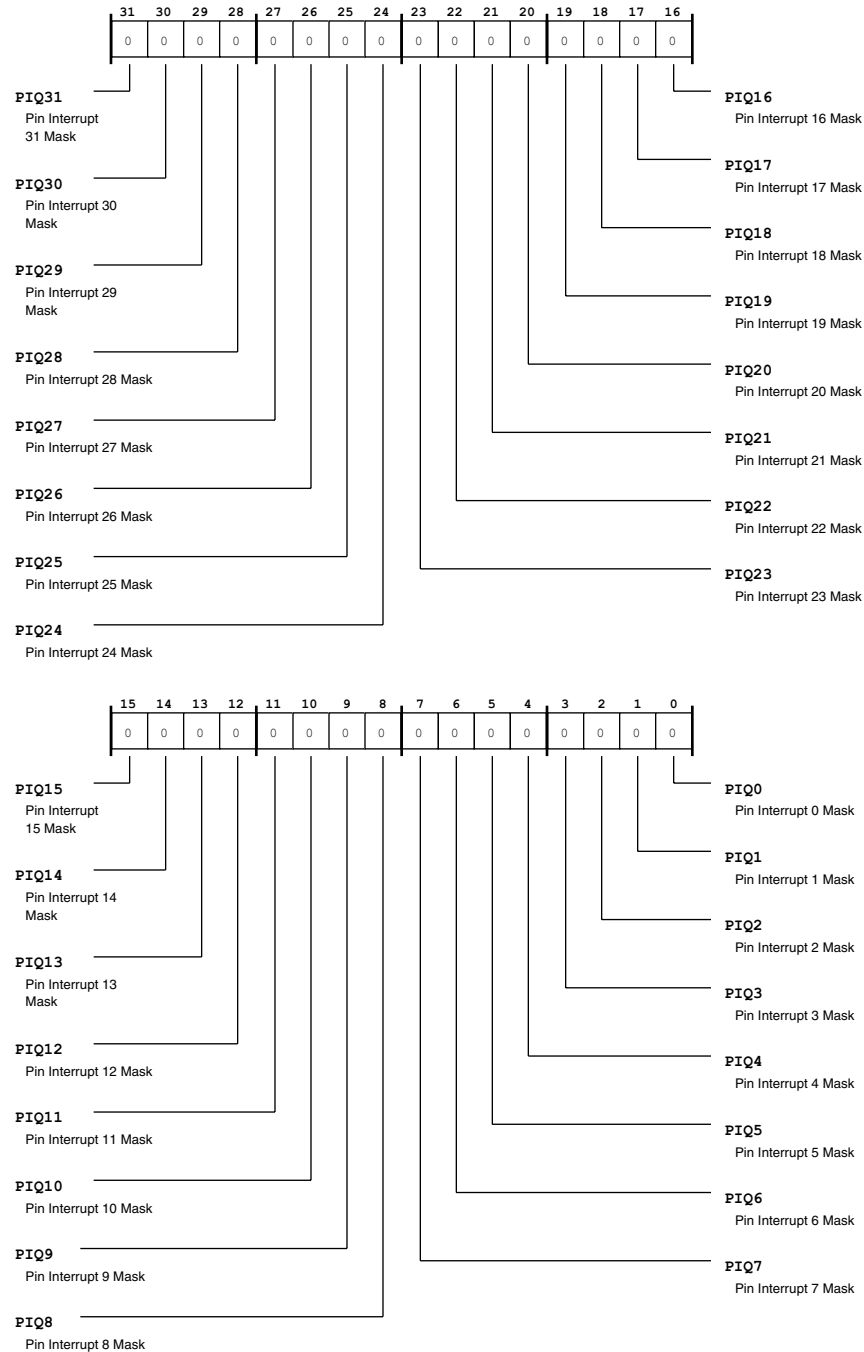


**Figure 14-27: PINT\_MSK\_CLR Register Diagram**

**Table 14-35:** PINT\_MSK\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Mask.<br>Set to disable interrupt. |
| 30      | PIQ30    | Pin Interrupt 30 Mask.<br>Set to disable interrupt. |
| 29      | PIQ29    | Pin Interrupt 29 Mask.<br>Set to disable interrupt. |
| 28      | PIQ28    | Pin Interrupt 28 Mask.<br>Set to disable interrupt. |
| 27      | PIQ27    | Pin Interrupt 27 Mask.<br>Set to disable interrupt. |
| 26      | PIQ26    | Pin Interrupt 26 Mask.<br>Set to disable interrupt. |
| 25      | PIQ25    | Pin Interrupt 25 Mask.<br>Set to disable interrupt. |
| 24      | PIQ24    | Pin Interrupt 24 Mask.<br>Set to disable interrupt. |
| 23      | PIQ23    | Pin Interrupt 23 Mask.<br>Set to disable interrupt. |
| 22      | PIQ22    | Pin Interrupt 22 Mask.<br>Set to disable interrupt. |
| 21      | PIQ21    | Pin Interrupt 21 Mask.<br>Set to disable interrupt. |
| 20      | PIQ20    | Pin Interrupt 20 Mask.<br>Set to disable interrupt. |
| 19      | PIQ19    | Pin Interrupt 19 Mask.<br>Set to disable interrupt. |
| 18      | PIQ18    | Pin Interrupt 18 Mask.<br>Set to disable interrupt. |



**Table 14-35:** PINT\_MSK\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Mask.<br>Set to disable interrupt. |
| 16      | PIQ16    | Pin Interrupt 16 Mask.<br>Set to disable interrupt. |
| 15      | PIQ15    | Pin Interrupt 15 Mask.<br>Set to disable interrupt. |
| 14      | PIQ14    | Pin Interrupt 14 Mask.<br>Set to disable interrupt. |
| 13      | PIQ13    | Pin Interrupt 13 Mask.<br>Set to disable interrupt. |
| 12      | PIQ12    | Pin Interrupt 12 Mask.<br>Set to disable interrupt. |
| 11      | PIQ11    | Pin Interrupt 11 Mask.<br>Set to disable interrupt. |
| 10      | PIQ10    | Pin Interrupt 10 Mask.<br>Set to disable interrupt. |
| 9       | PIQ9     | Pin Interrupt 9 Mask.<br>Set to disable interrupt.  |
| 8       | PIQ8     | Pin Interrupt 8 Mask.<br>Set to disable interrupt.  |
| 7       | PIQ7     | Pin Interrupt 7 Mask.<br>Set to disable interrupt.  |
| 6       | PIQ6     | Pin Interrupt 6 Mask.<br>Set to disable interrupt.  |
| 5       | PIQ5     | Pin Interrupt 5 Mask.<br>Set to disable interrupt.  |
| 4       | PIQ4     | Pin Interrupt 4 Mask.<br>Set to disable interrupt.  |

**Table 14-35:** PINT\_MSK\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Mask.<br>Set to disable interrupt. |
| 2       | PIQ2     | Pin Interrupt 2 Mask.<br>Set to disable interrupt. |
| 1       | PIQ1     | Pin Interrupt 1 Mask.<br>Set to disable interrupt. |
| 0       | PIQ0     | Pin Interrupt 0 Mask.<br>Set to disable interrupt. |

## Pint Request Register

The `PINT_REQ` register indicates interrupt request status for pin interrupts. When set, an interrupt request is pending. When cleared, there is no interrupt request pending.

Both the `PINT_REQ` and `PINT_LATCH` registers indicate whether an interrupt request is latched on the respective pin. The `PINT_LATCH` register is a latch that operates regardless of the interrupt masks. Bits of the `PINT_REQ` register depend on the mask register. The `PINT_REQ` register is a logical AND of the `PINT_LATCH` register and the interrupt mask.

PINT\_REQ: Pint Request Register - R/W

Reset = 0x0000 0000

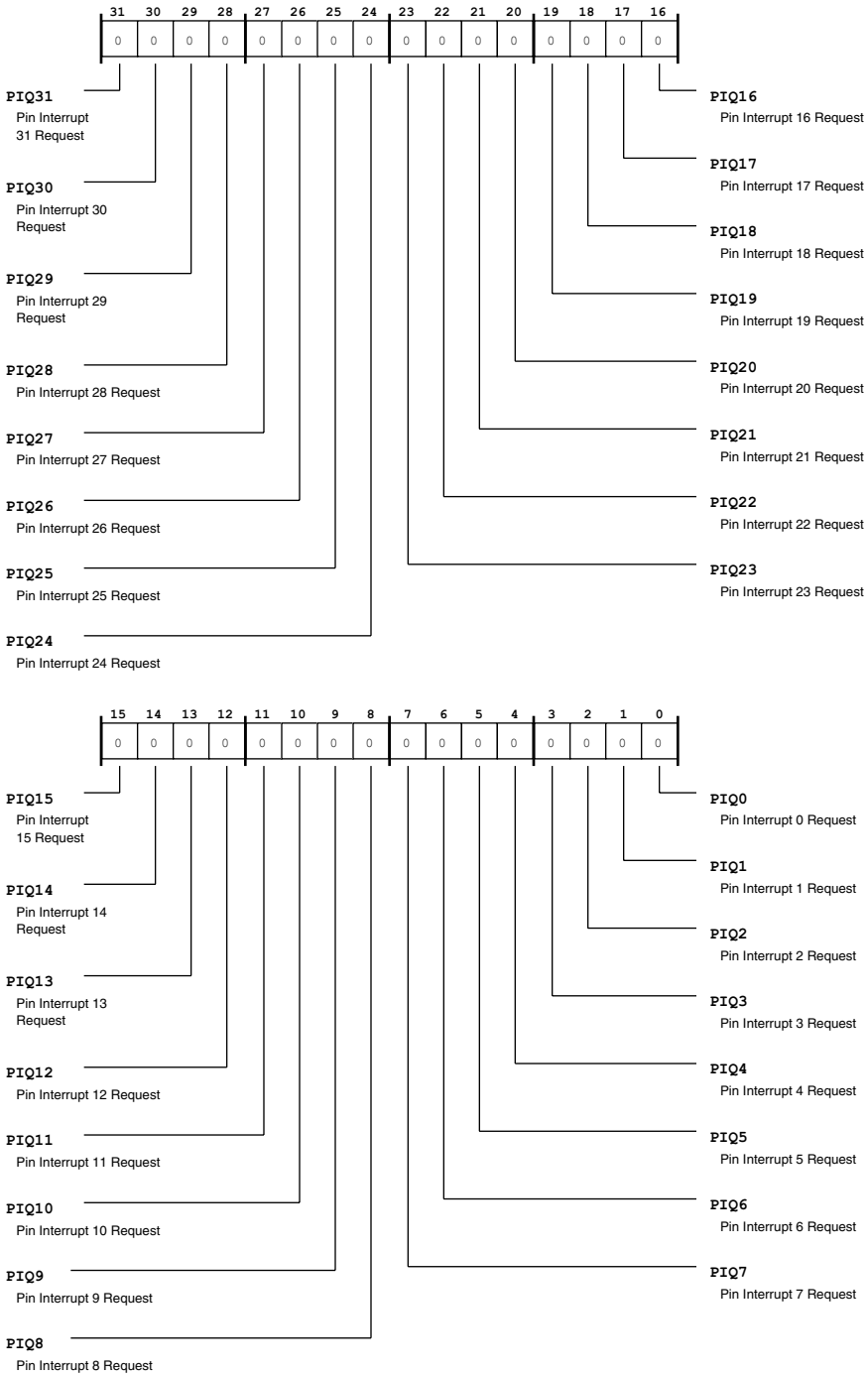


Figure 14-28: PINT\_REQ Register Diagram

**Table 14-36:** PINT\_REQ Register Fields

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Request.<br>If set, request pending. |
| 30      | PIQ30    | Pin Interrupt 30 Request.<br>If set, request pending. |
| 29      | PIQ29    | Pin Interrupt 29 Request.<br>If set, request pending. |
| 28      | PIQ28    | Pin Interrupt 28 Request.<br>If set, request pending. |
| 27      | PIQ27    | Pin Interrupt 27 Request.<br>If set, request pending. |
| 26      | PIQ26    | Pin Interrupt 26 Request.<br>If set, request pending. |
| 25      | PIQ25    | Pin Interrupt 25 Request.<br>If set, request pending. |
| 24      | PIQ24    | Pin Interrupt 24 Request.<br>If set, request pending. |
| 23      | PIQ23    | Pin Interrupt 23 Request.<br>If set, request pending. |
| 22      | PIQ22    | Pin Interrupt 22 Request.<br>If set, request pending. |
| 21      | PIQ21    | Pin Interrupt 21 Request.<br>If set, request pending. |
| 20      | PIQ20    | Pin Interrupt 20 Request.<br>If set, request pending. |
| 19      | PIQ19    | Pin Interrupt 19 Request.<br>If set, request pending. |
| 18      | PIQ18    | Pin Interrupt 18 Request.<br>If set, request pending. |

**Table 14-36:** PINT\_REQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Request.<br>If set, request pending. |
| 16      | PIQ16    | Pin Interrupt 16 Request.<br>If set, request pending. |
| 15      | PIQ15    | Pin Interrupt 15 Request.<br>If set, request pending. |
| 14      | PIQ14    | Pin Interrupt 14 Request.<br>If set, request pending. |
| 13      | PIQ13    | Pin Interrupt 13 Request.<br>If set, request pending. |
| 12      | PIQ12    | Pin Interrupt 12 Request.<br>If set, request pending. |
| 11      | PIQ11    | Pin Interrupt 11 Request.<br>If set, request pending. |
| 10      | PIQ10    | Pin Interrupt 10 Request.<br>If set, request pending. |
| 9       | PIQ9     | Pin Interrupt 9 Request.<br>If set, request pending.  |
| 8       | PIQ8     | Pin Interrupt 8 Request.<br>If set, request pending.  |
| 7       | PIQ7     | Pin Interrupt 7 Request.<br>If set, request pending.  |
| 6       | PIQ6     | Pin Interrupt 6 Request.<br>If set, request pending.  |
| 5       | PIQ5     | Pin Interrupt 5 Request.<br>If set, request pending.  |
| 4       | PIQ4     | Pin Interrupt 4 Request.<br>If set, request pending.  |

Table 14-36: PINT\_REQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                              |
|---------|----------|------------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Request.<br>If set, request pending. |
| 2       | PIQ2     | Pin Interrupt 2 Request.<br>If set, request pending. |
| 1       | PIQ1     | Pin Interrupt 1 Request.<br>If set, request pending. |
| 0       | PIQ0     | Pin Interrupt 0 Request.<br>If set, request pending. |

Pint Assign Register

The PINT\_ASSIGN register controls the pin-to-interrupt assignment in a byte-wide manner. This register consists of four control bytes that each function as a multiplexer control.

The PINT ports are subdivided into 8-bit half ports, resulting in lower and upper half 8-bit units. Using the multiplexers controlled by the PINT\_ASSIGN register, the lower half units of eight pins can be forwarded to either byte 0 or byte 2 of either associated PINT block, and the upper half units can be forwarded to either byte 1 or byte 3 of the PINT block, without further restrictions.

PINT\_ASSIGN: Pint Assign Register - R/W

Reset = 0x0000 0101

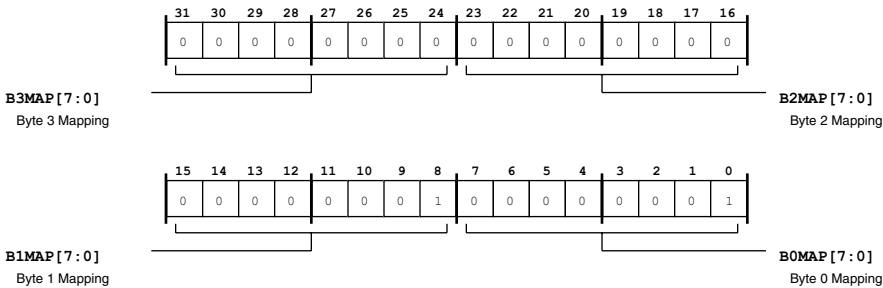


Figure 14-29: PINT\_ASSIGN Register Diagram

**Table 14-37:** PINT\_ASSIGN Register Fields

| Bit No. | Bit Name | Description/Enumeration |                            |
|---------|----------|-------------------------|----------------------------|
| 31:24   | B3MAP    | Byte 3 Mapping.         |                            |
|         |          | 0                       | B3MAP_PAH<br>Byte 3 = PA.H |
|         |          | 1                       | B3MAP_PBH<br>Byte 3 = PB.H |
| 23:16   | B2MAP    | Byte 2 Mapping.         |                            |
|         |          | 0                       | B2MAP_PAL<br>Byte 2 = PA.L |
|         |          | 1                       | B2MAP_PBL<br>Byte 2 = PB.L |
| 15:8    | B1MAP    | Byte 1 Mapping.         |                            |
|         |          | 0                       | B1MAP_PAH<br>Byte 1 = PA.H |
|         |          | 1                       | B1MAP_PBH<br>Byte 1 = PB.H |
| 7:0     | B0MAP    | Byte 0 Mapping.         |                            |
|         |          | 0                       | B0MAP_PAL<br>Byte 0 = PA.L |
|         |          | 1                       | B0MAP_PBL<br>Byte 0 = PB.L |

## Pint Edge Set Register

The PINT\_EDGE\_SET register permits selecting edge-sensitive interrupts. Writing 1 to a bit in PINT\_EDGE\_SET enables edge sensitivity for the corresponding pin interrupt.

PINT\_EDGE\_SET: Pint Edge Set Register - R/W

Reset = 0x0000 0000

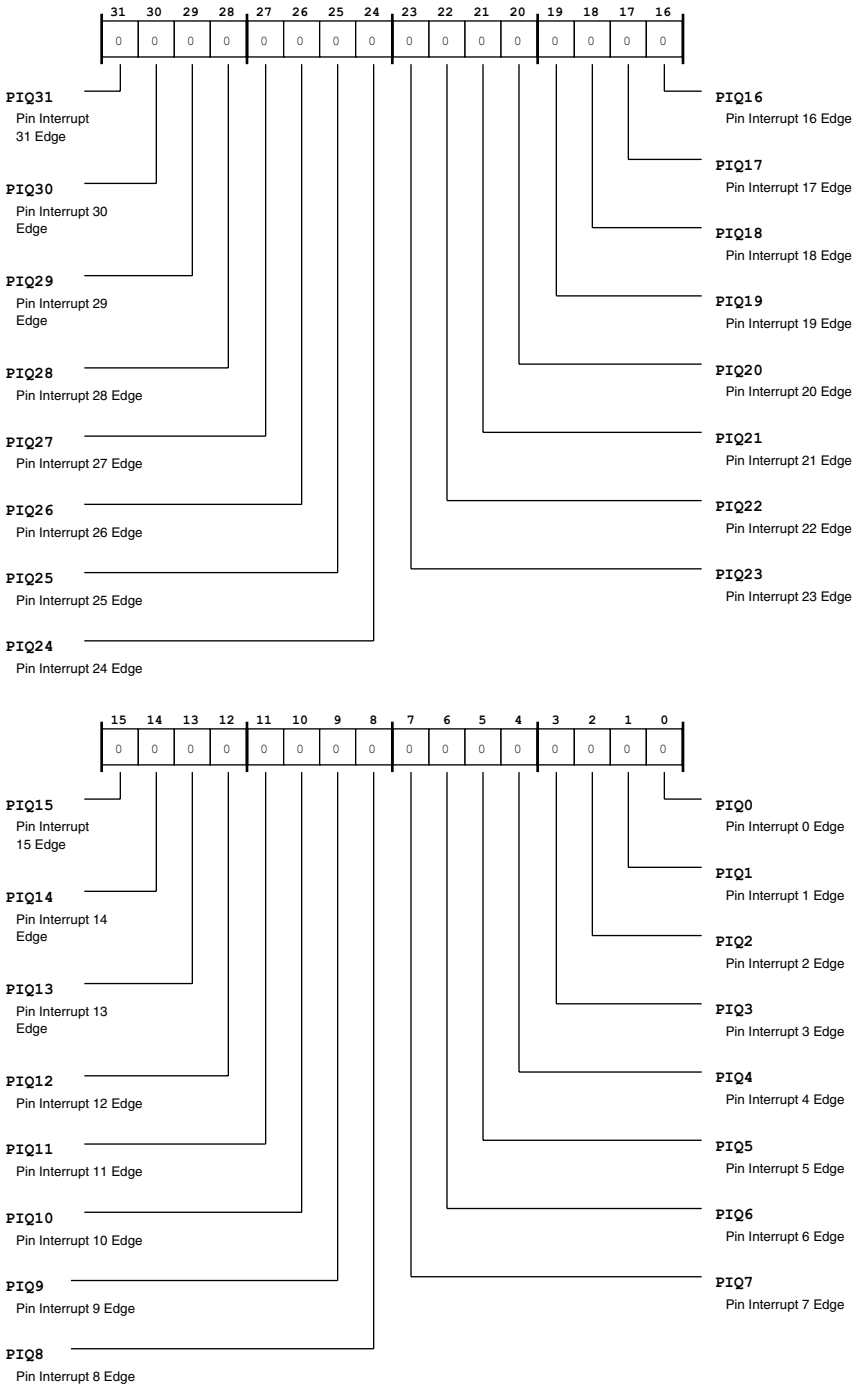


Figure 14-30: PINT\_EDGE\_SET Register Diagram



**Table 14-38:** PINT\_EDGE\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                                   |
|---------|----------|-----------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Edge.<br>Set to enable edge sensitivity. |
| 30      | PIQ30    | Pin Interrupt 30 Edge.<br>Set to enable edge sensitivity. |
| 29      | PIQ29    | Pin Interrupt 29 Edge.<br>Set to enable edge sensitivity. |
| 28      | PIQ28    | Pin Interrupt 28 Edge.<br>Set to enable edge sensitivity. |
| 27      | PIQ27    | Pin Interrupt 27 Edge.<br>Set to enable edge sensitivity. |
| 26      | PIQ26    | Pin Interrupt 26 Edge.<br>Set to enable edge sensitivity. |
| 25      | PIQ25    | Pin Interrupt 25 Edge.<br>Set to enable edge sensitivity. |
| 24      | PIQ24    | Pin Interrupt 24 Edge.<br>Set to enable edge sensitivity. |
| 23      | PIQ23    | Pin Interrupt 23 Edge.<br>Set to enable edge sensitivity. |
| 22      | PIQ22    | Pin Interrupt 22 Edge.<br>Set to enable edge sensitivity. |
| 21      | PIQ21    | Pin Interrupt 21 Edge.<br>Set to enable edge sensitivity. |
| 20      | PIQ20    | Pin Interrupt 20 Edge.<br>Set to enable edge sensitivity. |
| 19      | PIQ19    | Pin Interrupt 19 Edge.<br>Set to enable edge sensitivity. |
| 18      | PIQ18    | Pin Interrupt 18 Edge.<br>Set to enable edge sensitivity. |

**Table 14-38:** PINT\_EDGE\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                   |
|---------|----------|-----------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Edge.<br>Set to enable edge sensitivity. |
| 16      | PIQ16    | Pin Interrupt 16 Edge.<br>Set to enable edge sensitivity. |
| 15      | PIQ15    | Pin Interrupt 15 Edge.<br>Set to enable edge sensitivity. |
| 14      | PIQ14    | Pin Interrupt 14 Edge.<br>Set to enable edge sensitivity. |
| 13      | PIQ13    | Pin Interrupt 13 Edge.<br>Set to enable edge sensitivity. |
| 12      | PIQ12    | Pin Interrupt 12 Edge.<br>Set to enable edge sensitivity. |
| 11      | PIQ11    | Pin Interrupt 11 Edge.<br>Set to enable edge sensitivity. |
| 10      | PIQ10    | Pin Interrupt 10 Edge.<br>Set to enable edge sensitivity. |
| 9       | PIQ9     | Pin Interrupt 9 Edge.<br>Set to enable edge sensitivity.  |
| 8       | PIQ8     | Pin Interrupt 8 Edge.<br>Set to enable edge sensitivity.  |
| 7       | PIQ7     | Pin Interrupt 7 Edge.<br>Set to enable edge sensitivity.  |
| 6       | PIQ6     | Pin Interrupt 6 Edge.<br>Set to enable edge sensitivity.  |
| 5       | PIQ5     | Pin Interrupt 5 Edge.<br>Set to enable edge sensitivity.  |
| 4       | PIQ4     | Pin Interrupt 4 Edge.<br>Set to enable edge sensitivity.  |

**Table 14-38:** PINT\_EDGE\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                  |
|---------|----------|----------------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Edge.<br>Set to enable edge sensitivity. |
| 2       | PIQ2     | Pin Interrupt 2 Edge.<br>Set to enable edge sensitivity. |
| 1       | PIQ1     | Pin Interrupt 1 Edge.<br>Set to enable edge sensitivity. |
| 0       | PIQ0     | Pin Interrupt 0 Edge.<br>Set to enable edge sensitivity. |

## Pint Edge Clear Register

The PINT\_EDGE\_CLR register permits selecting level-sensitive interrupts. Writing 1 to a bit in PINT\_EDGE\_CLR enables level sensitivity for the corresponding pin interrupt.

PINT\_EDGE\_CLR: Pint Edge Clear Register - R/W

Reset = 0x0000 0000

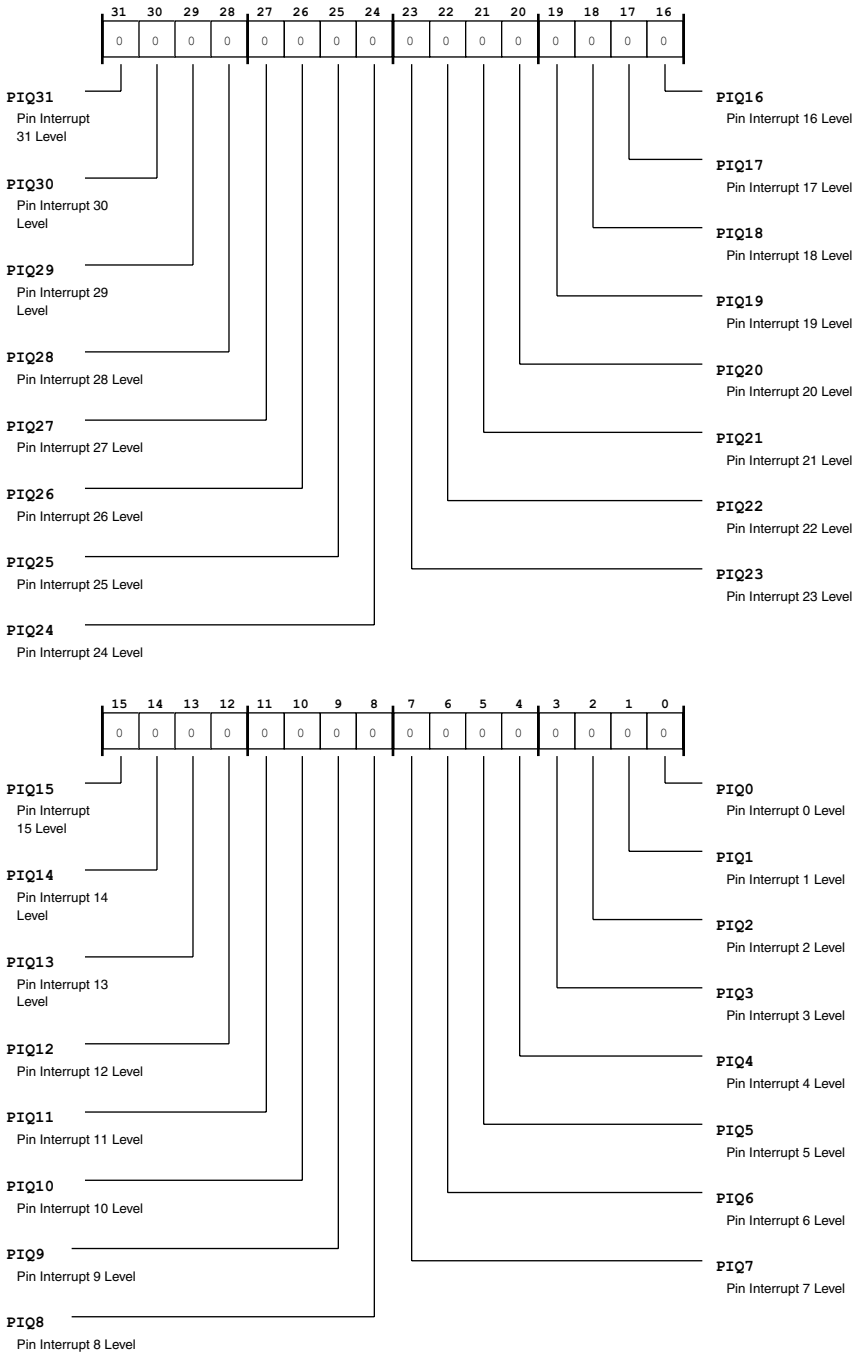


Figure 14-31: PINT\_EDGE\_CLR Register Diagram

**Table 14-39:** PINT\_EDGE\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                     |
|---------|----------|-------------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Level.<br>Set to enable level sensitivity. |
| 30      | PIQ30    | Pin Interrupt 30 Level.<br>Set to enable level sensitivity. |
| 29      | PIQ29    | Pin Interrupt 29 Level.<br>Set to enable level sensitivity. |
| 28      | PIQ28    | Pin Interrupt 28 Level.<br>Set to enable level sensitivity. |
| 27      | PIQ27    | Pin Interrupt 27 Level.<br>Set to enable level sensitivity. |
| 26      | PIQ26    | Pin Interrupt 26 Level.<br>Set to enable level sensitivity. |
| 25      | PIQ25    | Pin Interrupt 25 Level.<br>Set to enable level sensitivity. |
| 24      | PIQ24    | Pin Interrupt 24 Level.<br>Set to enable level sensitivity. |
| 23      | PIQ23    | Pin Interrupt 23 Level.<br>Set to enable level sensitivity. |
| 22      | PIQ22    | Pin Interrupt 22 Level.<br>Set to enable level sensitivity. |
| 21      | PIQ21    | Pin Interrupt 21 Level.<br>Set to enable level sensitivity. |
| 20      | PIQ20    | Pin Interrupt 20 Level.<br>Set to enable level sensitivity. |
| 19      | PIQ19    | Pin Interrupt 19 Level.<br>Set to enable level sensitivity. |
| 18      | PIQ18    | Pin Interrupt 18 Level.<br>Set to enable level sensitivity. |

**Table 14-39:** PINT\_EDGE\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                     |
|---------|----------|-------------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Level.<br>Set to enable level sensitivity. |
| 16      | PIQ16    | Pin Interrupt 16 Level.<br>Set to enable level sensitivity. |
| 15      | PIQ15    | Pin Interrupt 15 Level.<br>Set to enable level sensitivity. |
| 14      | PIQ14    | Pin Interrupt 14 Level.<br>Set to enable level sensitivity. |
| 13      | PIQ13    | Pin Interrupt 13 Level.<br>Set to enable level sensitivity. |
| 12      | PIQ12    | Pin Interrupt 12 Level.<br>Set to enable level sensitivity. |
| 11      | PIQ11    | Pin Interrupt 11 Level.<br>Set to enable level sensitivity. |
| 10      | PIQ10    | Pin Interrupt 10 Level.<br>Set to enable level sensitivity. |
| 9       | PIQ9     | Pin Interrupt 9 Level.<br>Set to enable level sensitivity.  |
| 8       | PIQ8     | Pin Interrupt 8 Level.<br>Set to enable level sensitivity.  |
| 7       | PIQ7     | Pin Interrupt 7 Level.<br>Set to enable level sensitivity.  |
| 6       | PIQ6     | Pin Interrupt 6 Level.<br>Set to enable level sensitivity.  |
| 5       | PIQ5     | Pin Interrupt 5 Level.<br>Set to enable level sensitivity.  |
| 4       | PIQ4     | Pin Interrupt 4 Level.<br>Set to enable level sensitivity.  |

**Table 14-39:** PINT\_EDGE\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                    |
|---------|----------|------------------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Level.<br>Set to enable level sensitivity. |
| 2       | PIQ2     | Pin Interrupt 2 Level.<br>Set to enable level sensitivity. |
| 1       | PIQ1     | Pin Interrupt 1 Level.<br>Set to enable level sensitivity. |
| 0       | PIQ0     | Pin Interrupt 0 Level.<br>Set to enable level sensitivity. |

## Pint Invert Set Register

The `PINT_INV_SET` register enables inverting input polarity. Writing 1 to a bit in `PINT_INV_SET` enables an inverter for input on the corresponding pin.

PINT\_INV\_SET: Pint Invert Set Register - R/W

Reset = 0x0000 0000

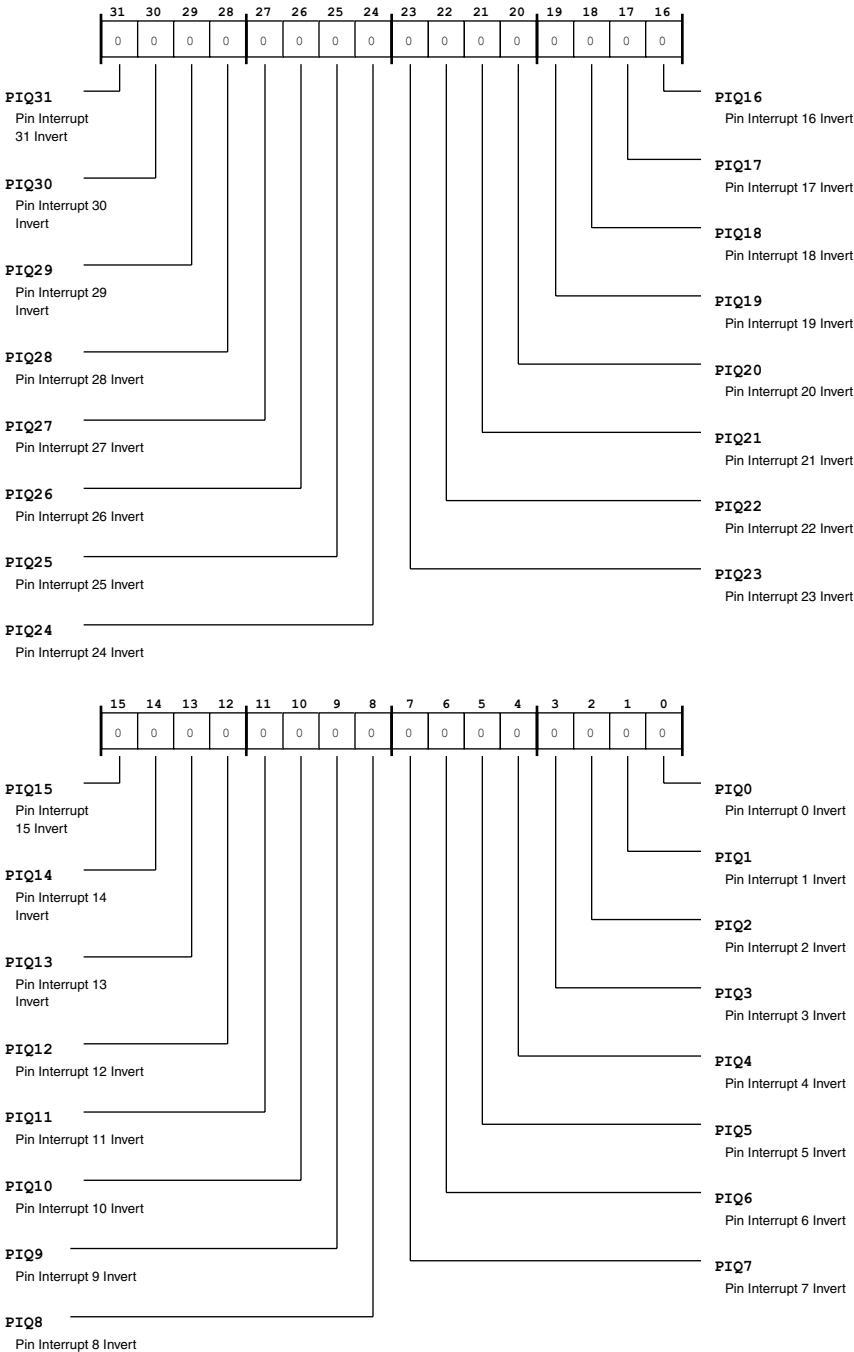


Figure 14-32: PINT\_INV\_SET Register Diagram



**Table 14-40:** PINT\_INV\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                                   |
|---------|----------|-----------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Invert.<br>Set to enable inverted input. |
| 30      | PIQ30    | Pin Interrupt 30 Invert.<br>Set to enable inverted input. |
| 29      | PIQ29    | Pin Interrupt 29 Invert.<br>Set to enable inverted input. |
| 28      | PIQ28    | Pin Interrupt 28 Invert.<br>Set to enable inverted input. |
| 27      | PIQ27    | Pin Interrupt 27 Invert.<br>Set to enable inverted input. |
| 26      | PIQ26    | Pin Interrupt 26 Invert.<br>Set to enable inverted input. |
| 25      | PIQ25    | Pin Interrupt 25 Invert.<br>Set to enable inverted input. |
| 24      | PIQ24    | Pin Interrupt 24 Invert.<br>Set to enable inverted input. |
| 23      | PIQ23    | Pin Interrupt 23 Invert.<br>Set to enable inverted input. |
| 22      | PIQ22    | Pin Interrupt 22 Invert.<br>Set to enable inverted input. |
| 21      | PIQ21    | Pin Interrupt 21 Invert.<br>Set to enable inverted input. |
| 20      | PIQ20    | Pin Interrupt 20 Invert.<br>Set to enable inverted input. |
| 19      | PIQ19    | Pin Interrupt 19 Invert.<br>Set to enable inverted input. |
| 18      | PIQ18    | Pin Interrupt 18 Invert.<br>Set to enable inverted input. |

**Table 14-40:** PINT\_INV\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                   |
|---------|----------|-----------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Invert.<br>Set to enable inverted input. |
| 16      | PIQ16    | Pin Interrupt 16 Invert.<br>Set to enable inverted input. |
| 15      | PIQ15    | Pin Interrupt 15 Invert.<br>Set to enable inverted input. |
| 14      | PIQ14    | Pin Interrupt 14 Invert.<br>Set to enable inverted input. |
| 13      | PIQ13    | Pin Interrupt 13 Invert.<br>Set to enable inverted input. |
| 12      | PIQ12    | Pin Interrupt 12 Invert.<br>Set to enable inverted input. |
| 11      | PIQ11    | Pin Interrupt 11 Invert.<br>Set to enable inverted input. |
| 10      | PIQ10    | Pin Interrupt 10 Invert.<br>Set to enable inverted input. |
| 9       | PIQ9     | Pin Interrupt 9 Invert.<br>Set to enable inverted input.  |
| 8       | PIQ8     | Pin Interrupt 8 Invert.<br>Set to enable inverted input.  |
| 7       | PIQ7     | Pin Interrupt 7 Invert.<br>Set to enable inverted input.  |
| 6       | PIQ6     | Pin Interrupt 6 Invert.<br>Set to enable inverted input.  |
| 5       | PIQ5     | Pin Interrupt 5 Invert.<br>Set to enable inverted input.  |
| 4       | PIQ4     | Pin Interrupt 4 Invert.<br>Set to enable inverted input.  |

**Table 14-40:** PINT\_INV\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                  |
|---------|----------|----------------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Invert.<br>Set to enable inverted input. |
| 2       | PIQ2     | Pin Interrupt 2 Invert.<br>Set to enable inverted input. |
| 1       | PIQ1     | Pin Interrupt 1 Invert.<br>Set to enable inverted input. |
| 0       | PIQ0     | Pin Interrupt 0 Invert.<br>Set to enable inverted input. |

## Pint Invert Clear Register

The `PINT_INV_CLR` register disables inverting input polarity. Writing 1 to a bit in `PINT_INV_CLR` disables an inverter for input on the corresponding pin.

PINT\_INV\_CLR: Pint Invert Clear Register - R/W

Reset = 0x0000 0000

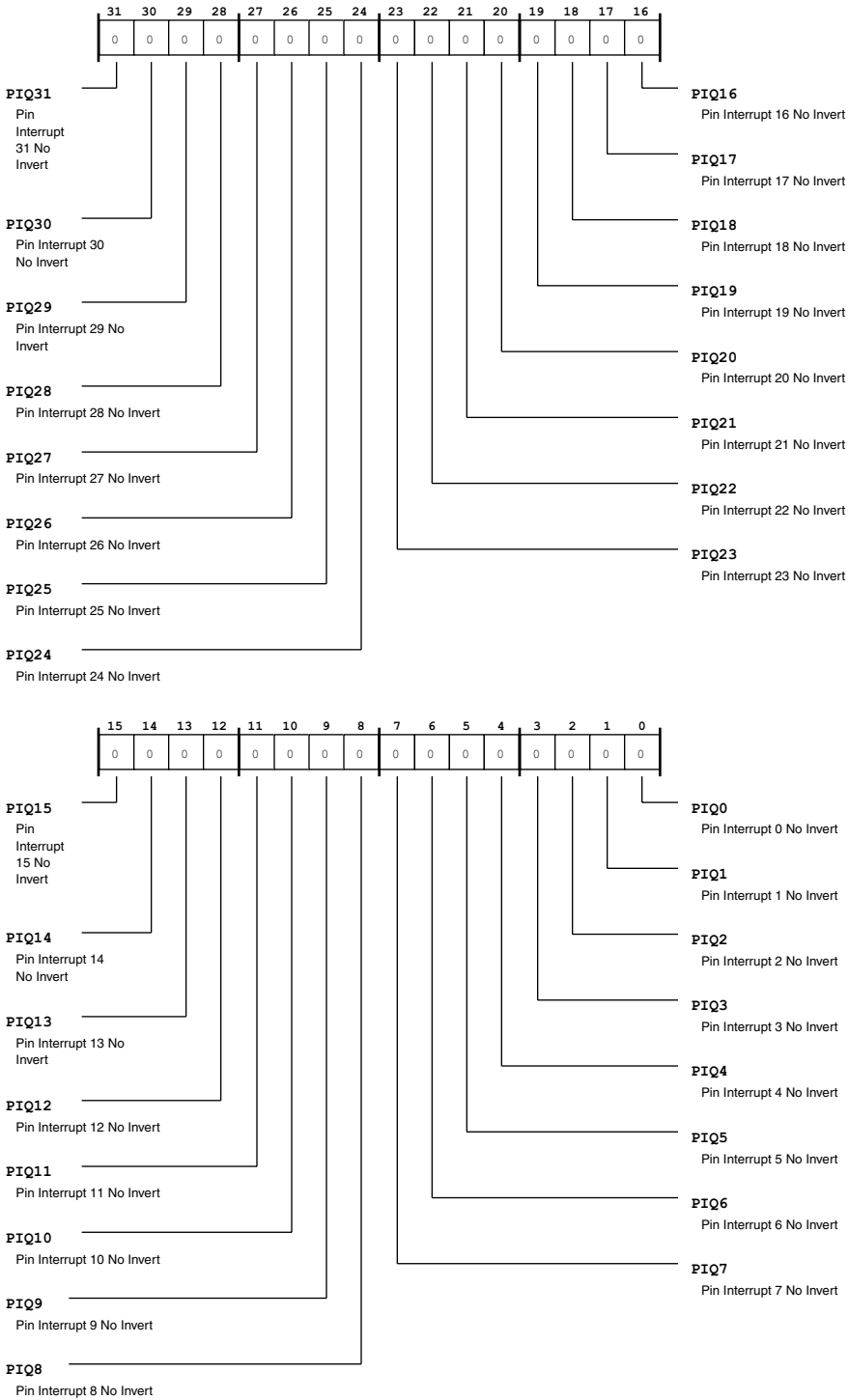


Figure 14-33: PINT\_INV\_CLR Register Diagram

**Table 14-41:** PINT\_INV\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                       |
|---------|----------|---------------------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 No Invert.<br>Set to disable inverted input. |
| 30      | PIQ30    | Pin Interrupt 30 No Invert.<br>Set to disable inverted input. |
| 29      | PIQ29    | Pin Interrupt 29 No Invert.<br>Set to disable inverted input. |
| 28      | PIQ28    | Pin Interrupt 28 No Invert.<br>Set to disable inverted input. |
| 27      | PIQ27    | Pin Interrupt 27 No Invert.<br>Set to disable inverted input. |
| 26      | PIQ26    | Pin Interrupt 26 No Invert.<br>Set to disable inverted input. |
| 25      | PIQ25    | Pin Interrupt 25 No Invert.<br>Set to disable inverted input. |
| 24      | PIQ24    | Pin Interrupt 24 No Invert.<br>Set to disable inverted input. |
| 23      | PIQ23    | Pin Interrupt 23 No Invert.<br>Set to disable inverted input. |
| 22      | PIQ22    | Pin Interrupt 22 No Invert.<br>Set to disable inverted input. |
| 21      | PIQ21    | Pin Interrupt 21 No Invert.<br>Set to disable inverted input. |
| 20      | PIQ20    | Pin Interrupt 20 No Invert.<br>Set to disable inverted input. |
| 19      | PIQ19    | Pin Interrupt 19 No Invert.<br>Set to disable inverted input. |
| 18      | PIQ18    | Pin Interrupt 18 No Invert.<br>Set to disable inverted input. |

**Table 14-41:** PINT\_INV\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                       |
|---------|----------|---------------------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 No Invert.<br>Set to disable inverted input. |
| 16      | PIQ16    | Pin Interrupt 16 No Invert.<br>Set to disable inverted input. |
| 15      | PIQ15    | Pin Interrupt 15 No Invert.<br>Set to disable inverted input. |
| 14      | PIQ14    | Pin Interrupt 14 No Invert.<br>Set to disable inverted input. |
| 13      | PIQ13    | Pin Interrupt 13 No Invert.<br>Set to disable inverted input. |
| 12      | PIQ12    | Pin Interrupt 12 No Invert.<br>Set to disable inverted input. |
| 11      | PIQ11    | Pin Interrupt 11 No Invert.<br>Set to disable inverted input. |
| 10      | PIQ10    | Pin Interrupt 10 No Invert.<br>Set to disable inverted input. |
| 9       | PIQ9     | Pin Interrupt 9 No Invert.<br>Set to disable inverted input.  |
| 8       | PIQ8     | Pin Interrupt 8 No Invert.<br>Set to disable inverted input.  |
| 7       | PIQ7     | Pin Interrupt 7 No Invert.<br>Set to disable inverted input.  |
| 6       | PIQ6     | Pin Interrupt 6 No Invert.<br>Set to disable inverted input.  |
| 5       | PIQ5     | Pin Interrupt 5 No Invert.<br>Set to disable inverted input.  |
| 4       | PIQ4     | Pin Interrupt 4 No Invert.<br>Set to disable inverted input.  |

**Table 14-41:** PINT\_INV\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                      |
|---------|----------|--------------------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 No Invert.<br>Set to disable inverted input. |
| 2       | PIQ2     | Pin Interrupt 2 No Invert.<br>Set to disable inverted input. |
| 1       | PIQ1     | Pin Interrupt 1 No Invert.<br>Set to disable inverted input. |
| 0       | PIQ0     | Pin Interrupt 0 No Invert.<br>Set to disable inverted input. |

## Pint Pinstate Register

When a half port is assigned to a byte in any PINT block, the state of the eight pins (regardless of GPIO or function, input or output) can be seen in the `PINT_PINSTATE` register. While neither input nor output drivers of the pin are enabled, reads of the pin state in `PINT_PINSTATE` return zero. The `PINT_PINSTATE` register reports the inverted state of the pin if the signal inverter is activated by the `PINT_INV_SET` register. The inverter can be enabled on a individual bit by bit basis. Every bit in the `PINT_INV_SET` and `PINT_INV_CLR` register pair represents a pin signal.

The pin interrupt pin state registers enable the service routine to read the current state of the pin without reading from GPIO space. If there was an edge-sensitive interrupt, the service routine can check whether the state of the pin is still high or turned low.

PINT\_PINSTATE: Pint Pinstate Register - R/WE

Reset = 0x0000 0000

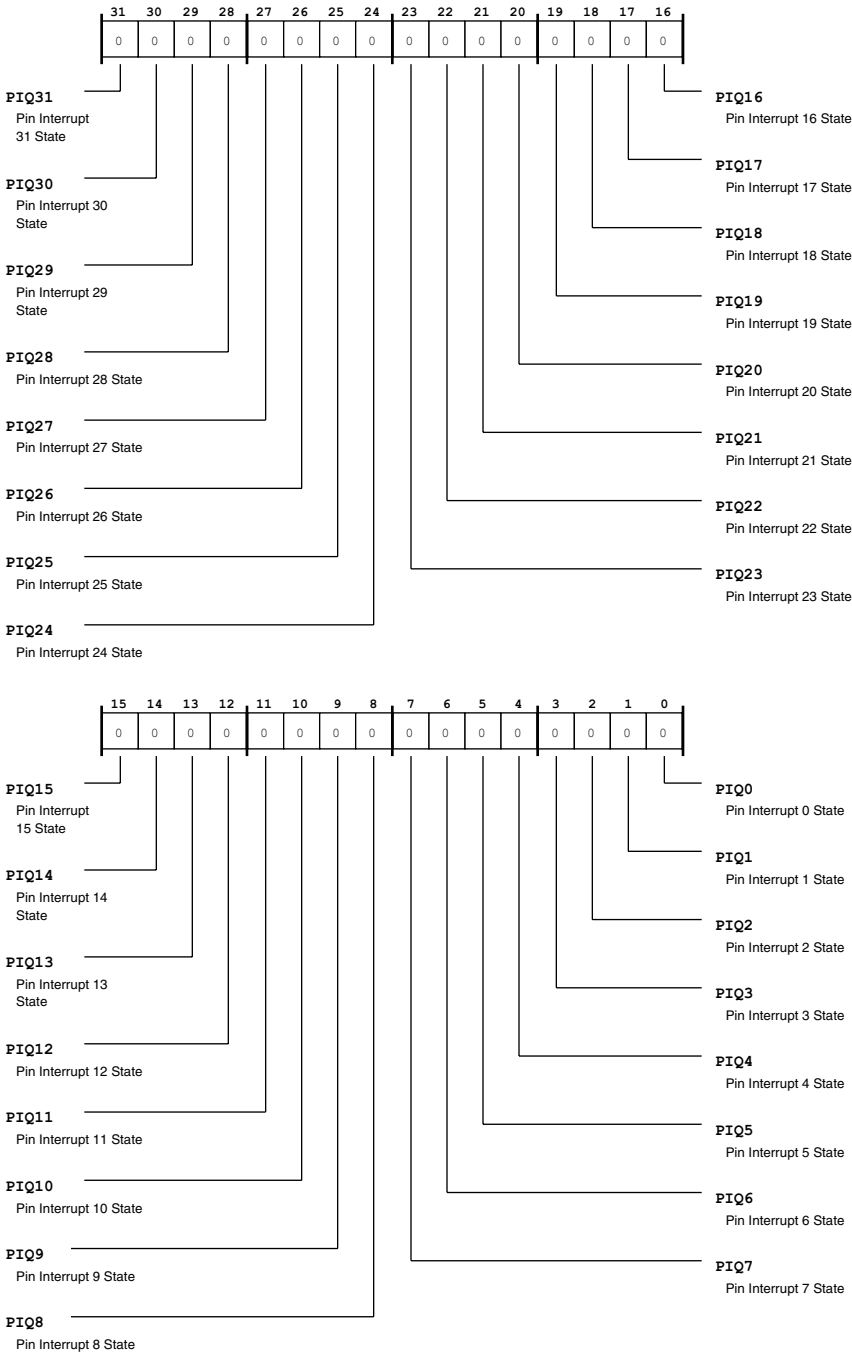


Figure 14-34: PINT\_PINSTATE Register Diagram



**Table 14-42:** PINT\_PINSTATE Register Fields

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 State.<br>Read returns pin state. |
| 30      | PIQ30    | Pin Interrupt 30 State.<br>Read returns pin state. |
| 29      | PIQ29    | Pin Interrupt 29 State.<br>Read returns pin state. |
| 28      | PIQ28    | Pin Interrupt 28 State.<br>Read returns pin state. |
| 27      | PIQ27    | Pin Interrupt 27 State.<br>Read returns pin state. |
| 26      | PIQ26    | Pin Interrupt 26 State.<br>Read returns pin state. |
| 25      | PIQ25    | Pin Interrupt 25 State.<br>Read returns pin state. |
| 24      | PIQ24    | Pin Interrupt 24 State.<br>Read returns pin state. |
| 23      | PIQ23    | Pin Interrupt 23 State.<br>Read returns pin state. |
| 22      | PIQ22    | Pin Interrupt 22 State.<br>Read returns pin state. |
| 21      | PIQ21    | Pin Interrupt 21 State.<br>Read returns pin state. |
| 20      | PIQ20    | Pin Interrupt 20 State.<br>Read returns pin state. |
| 19      | PIQ19    | Pin Interrupt 19 State.<br>Read returns pin state. |
| 18      | PIQ18    | Pin Interrupt 18 State.<br>Read returns pin state. |

**Table 14-42:** PINT\_PINSTATE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 State.<br>Read returns pin state. |
| 16      | PIQ16    | Pin Interrupt 16 State.<br>Read returns pin state. |
| 15      | PIQ15    | Pin Interrupt 15 State.<br>Read returns pin state. |
| 14      | PIQ14    | Pin Interrupt 14 State.<br>Read returns pin state. |
| 13      | PIQ13    | Pin Interrupt 13 State.<br>Read returns pin state. |
| 12      | PIQ12    | Pin Interrupt 12 State.<br>Read returns pin state. |
| 11      | PIQ11    | Pin Interrupt 11 State.<br>Read returns pin state. |
| 10      | PIQ10    | Pin Interrupt 10 State.<br>Read returns pin state. |
| 9       | PIQ9     | Pin Interrupt 9 State.<br>Read returns pin state.  |
| 8       | PIQ8     | Pin Interrupt 8 State.<br>Read returns pin state.  |
| 7       | PIQ7     | Pin Interrupt 7 State.<br>Read returns pin state.  |
| 6       | PIQ6     | Pin Interrupt 6 State.<br>Read returns pin state.  |
| 5       | PIQ5     | Pin Interrupt 5 State.<br>Read returns pin state.  |
| 4       | PIQ4     | Pin Interrupt 4 State.<br>Read returns pin state.  |

**Table 14-42:** PINT\_PINSTATE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                           |
|---------|----------|---------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 State.<br>Read returns pin state. |
| 2       | PIQ2     | Pin Interrupt 2 State.<br>Read returns pin state. |
| 1       | PIQ1     | Pin Interrupt 1 State.<br>Read returns pin state. |
| 0       | PIQ0     | Pin Interrupt 0 State.<br>Read returns pin state. |

## Pint Latch Register

The `PINT_LATCH` register indicates interrupt latch status for pin interrupts. When set, an interrupt request is latched. When cleared, there is no interrupt request latched.

Both the `PINT_REQ` and `PINT_LATCH` registers indicate whether an interrupt request is latched on the respective pin. The `PINT_LATCH` register is a latch that operates regardless of the interrupt masks. Bits of the `PINT_REQ` register depend on the mask register. The `PINT_REQ` register is a logical AND of the `PINT_LATCH` register and the interrupt mask.

Having two separate registers here enables the user to interrogate certain pins in polling mode while others work in interrupt mode. The `PINT_LATCH` registers can be used for edge detection or pin activity detection.

Both registers have W1C behavior. Writing a 1 to either clears respective bits in both registers. For interrupt operation, the user may prefer to W1C the `PINT_REQ` register (address still loaded in Px pointer). In polling mode it might be cleaner to W1C the `PINT_LATCH` register.

Regardless whether in edge-sensitive mode or level-sensitive mode, `PINT_LATCH` bits are never cleared by hardware except at system reset. Even in level-sensitive mode, the `PINT_LATCH` register functions as latch.

PINT\_LATCH: Pint Latch Register - R/W

Reset = 0x0000 0000

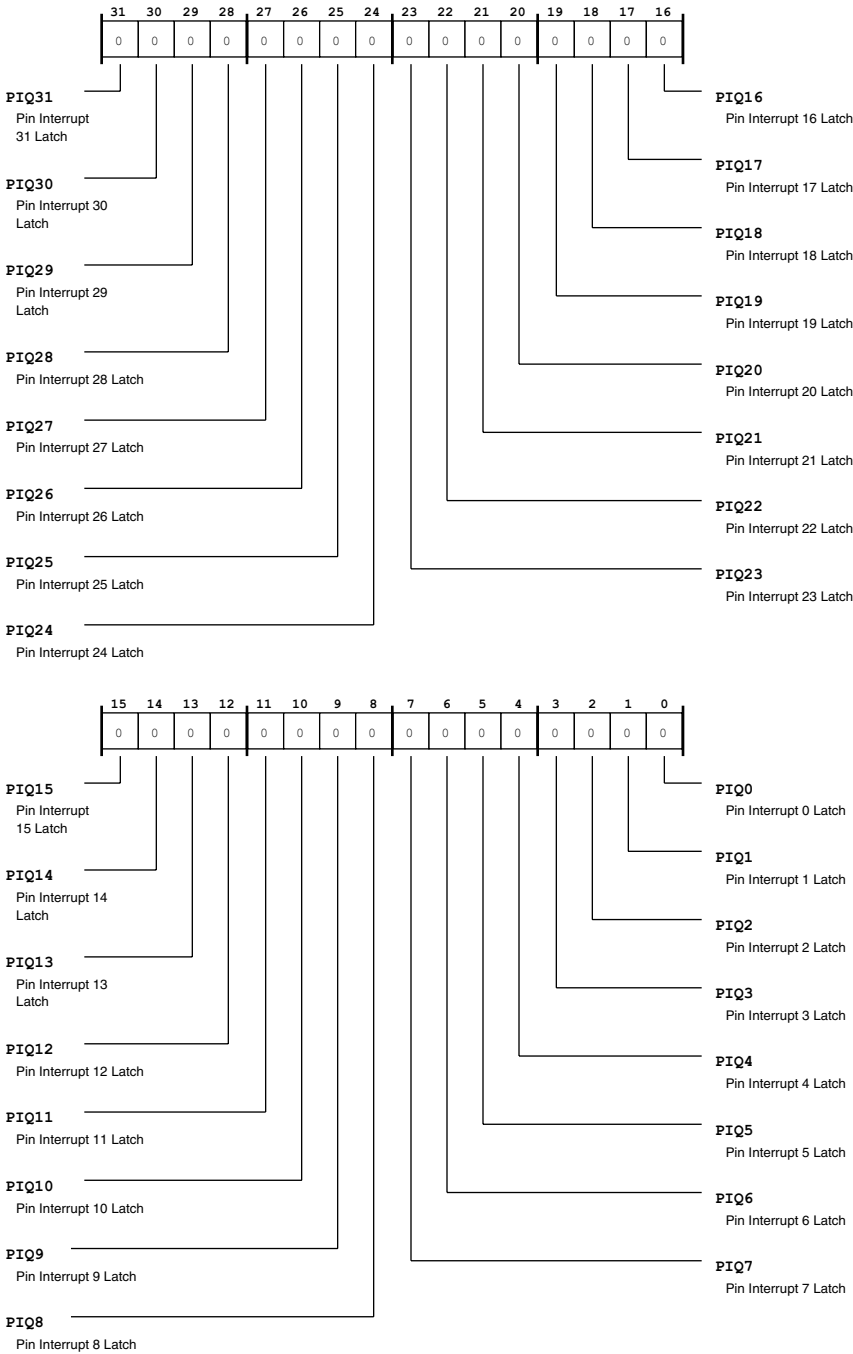


Figure 14-35: PINT\_LATCH Register Diagram

**Table 14-43:** PINT\_LATCH Register Fields

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 31      | PIQ31    | Pin Interrupt 31 Latch.<br>If set, request latched. |
| 30      | PIQ30    | Pin Interrupt 30 Latch.<br>If set, request latched. |
| 29      | PIQ29    | Pin Interrupt 29 Latch.<br>If set, request latched. |
| 28      | PIQ28    | Pin Interrupt 28 Latch.<br>If set, request latched. |
| 27      | PIQ27    | Pin Interrupt 27 Latch.<br>If set, request latched. |
| 26      | PIQ26    | Pin Interrupt 26 Latch.<br>If set, request latched. |
| 25      | PIQ25    | Pin Interrupt 25 Latch.<br>If set, request latched. |
| 24      | PIQ24    | Pin Interrupt 24 Latch.<br>If set, request latched. |
| 23      | PIQ23    | Pin Interrupt 23 Latch.<br>If set, request latched. |
| 22      | PIQ22    | Pin Interrupt 22 Latch.<br>If set, request latched. |
| 21      | PIQ21    | Pin Interrupt 21 Latch.<br>If set, request latched. |
| 20      | PIQ20    | Pin Interrupt 20 Latch.<br>If set, request latched. |
| 19      | PIQ19    | Pin Interrupt 19 Latch.<br>If set, request latched. |
| 18      | PIQ18    | Pin Interrupt 18 Latch.<br>If set, request latched. |

**Table 14-43:** PINT\_LATCH Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 17      | PIQ17    | Pin Interrupt 17 Latch.<br>If set, request latched. |
| 16      | PIQ16    | Pin Interrupt 16 Latch.<br>If set, request latched. |
| 15      | PIQ15    | Pin Interrupt 15 Latch.<br>If set, request latched. |
| 14      | PIQ14    | Pin Interrupt 14 Latch.<br>If set, request latched. |
| 13      | PIQ13    | Pin Interrupt 13 Latch.<br>If set, request latched. |
| 12      | PIQ12    | Pin Interrupt 12 Latch.<br>If set, request latched. |
| 11      | PIQ11    | Pin Interrupt 11 Latch.<br>If set, request latched. |
| 10      | PIQ10    | Pin Interrupt 10 Latch.<br>If set, request latched. |
| 9       | PIQ9     | Pin Interrupt 9 Latch.<br>If set, request latched.  |
| 8       | PIQ8     | Pin Interrupt 8 Latch.<br>If set, request latched.  |
| 7       | PIQ7     | Pin Interrupt 7 Latch.<br>If set, request latched.  |
| 6       | PIQ6     | Pin Interrupt 6 Latch.<br>If set, request latched.  |
| 5       | PIQ5     | Pin Interrupt 5 Latch.<br>If set, request latched.  |
| 4       | PIQ4     | Pin Interrupt 4 Latch.<br>If set, request latched.  |

**Table 14-43:** PINT\_LATCH Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 3       | PIQ3     | Pin Interrupt 3 Latch.<br>If set, request latched. |
| 2       | PIQ2     | Pin Interrupt 2 Latch.<br>If set, request latched. |
| 1       | PIQ1     | Pin Interrupt 1 Latch.<br>If set, request latched. |
| 0       | PIQ0     | Pin Interrupt 0 Latch.<br>If set, request latched. |





# 15 General-Purpose Timer (TIMER)

The general-purpose timer (TIMER) module serves as a collection of system timers that support various system-level functions. These functions include synchronized PWM waveform output capability, external signal capture, external event count, and general time base functionality. Additionally, a variety of interrupts can be generated upon completion of timer events. Moreover, GP timers can act both as trigger masters and trigger slaves.

## TIMER Features

Each timer can be individually configured in any of these modes:

- Pin interrupt capture mode
- Windowed Watchdog mode
- Pulse-width Count and Capture (WDTH\_CAP) mode
- External Event (EXT\_CLK) mode
- Pulse-width Modulation (PWM\_OUT) mode

Other features include:

- Synchronous operation
- Consistent management of period and pulse width values
- Autobaud detection for UART module (where available)
- Graceful bit pattern termination when stopping
- Support for center-aligned PWM patterns
- Error detection on implausible pattern values
- All read and write accesses to 32-bit registers are atomic
- Every timer has its dedicated interrupt request output
- Unused timers can function as edge-sensitive pin interrupts

**NOTE:**

Each timer has a `EMURUN` bit in its `TIMER_TMRn_CFG` register which controls whether to run or stop the timer during emulation. The emulation event is controlled by the SDU (System Debug Unit). Please refer to the SDU chapter for more details on generation of an emulation event.

## TIMER Register List

**Table 15-1:** TIMER Module Registers

| Name               | Description                       |
|--------------------|-----------------------------------|
| TIMER_REVID        | Revision ID Register              |
| TIMER_RUN          | Run Register                      |
| TIMER_RUN_SET      | Run Set Register                  |
| TIMER_RUN_CLR      | Run Clear Register                |
| TIMER_STOP_CFG     | Stop Configuration Register       |
| TIMER_STOP_CFG_SET | Stop Configuration Set Register   |
| TIMER_STOP_CFG_CLR | Stop Configuration Clear Register |
| TIMER_DATA_IMSK    | Data Interrupt Mask Register      |
| TIMER_STAT_IMSK    | Status Interrupt Mask Register    |
| TIMER_TRG_MSK      | Trigger Master Mask Register      |
| TIMER_TRG_IE       | Trigger Slave Enable Register     |
| TIMER_DATA_ILAT    | Data Interrupt Latch Register     |
| TIMER_STAT_ILAT    | Status Interrupt Latch Register   |
| TIMER_ERR_TYPE     | Error Type Status Register        |
| TIMER_BCAST_PER    | Broadcast Period Register         |
| TIMER_BCAST_WID    | Broadcast Width Register          |
| TIMER_BCAST_DLY    | Broadcast Delay Register          |

**Table 15-1:** TIMER Module Registers (Continued)

| Name           | Description                    |
|----------------|--------------------------------|
| TIMER_TMRn_CFG | Timer n Configuration Register |
| TIMER_TMRn_CNT | Timer n Counter Register       |
| TIMER_TMRn_PER | Timer n Period Register        |
| TIMER_TMRn_WID | Timer n Width Register         |
| TIMER_TMRn_DLY | Timer n Delay Register         |

## ADSP-BF60x TIMER Interrupt List

**Table 15-2:** ADSP-BF60x TIMER Interrupt List

| Description    | ID | DMA Channel | Sensitivity |
|----------------|----|-------------|-------------|
| TIMER0 Timer 0 | 12 |             | LEVEL       |
| TIMER0 Timer 1 | 13 |             | LEVEL       |
| TIMER0 Timer 2 | 14 |             | LEVEL       |
| TIMER0 Timer 3 | 15 |             | LEVEL       |
| TIMER0 Timer 4 | 16 |             | LEVEL       |
| TIMER0 Timer 5 | 17 |             | LEVEL       |
| TIMER0 Timer 6 | 18 |             | LEVEL       |
| TIMER0 Timer 7 | 19 |             | LEVEL       |
| TIMER0 Status  | 20 |             | LEVEL       |

## ADSP-BF60x TIMER Trigger List

**Table 15-3:** Masters ADSP-BF60x TIMER Trigger List

| Description    | ID | Sensitivity |
|----------------|----|-------------|
| TIMER0 Timer 0 | 2  | PULSE/EDGE  |

**Table 15-3:** Masters ADSP-BF60x TIMER Trigger List (Continued)

| Description    | ID | Sensitivity |
|----------------|----|-------------|
| TIMER0 Timer 1 | 3  | PULSE/EDGE  |
| TIMER0 Timer 2 | 4  | PULSE/EDGE  |
| TIMER0 Timer 3 | 5  | PULSE/EDGE  |
| TIMER0 Timer 4 | 6  | PULSE/EDGE  |
| TIMER0 Timer 5 | 7  | PULSE/EDGE  |
| TIMER0 Timer 6 | 8  | PULSE/EDGE  |
| TIMER0 Timer 7 | 9  | PULSE/EDGE  |
|                |    |             |

**Table 15-4:** Slaves ADSP-BF60x TIMER Trigger List

| Description    | ID | Sensitivity |
|----------------|----|-------------|
| TIMER0 Timer 0 | 2  |             |
| TIMER0 Timer 1 | 3  |             |
| TIMER0 Timer 2 | 4  |             |
| TIMER0 Timer 3 | 5  |             |
| TIMER0 Timer 4 | 6  |             |
| TIMER0 Timer 5 | 7  |             |
| TIMER0 Timer 6 | 8  |             |
| TIMER0 Timer 7 | 9  |             |
|                |    |             |

## TIMER Internal Interface

Timer registers are always accessed by the processor core through the MMR access bus. Hardware ensures that all read and write operations from and to 32-bit timer registers are atomic. Every timer has its dedicated data interrupt request. There is also one common timer status/error interrupt request output that connects to the System Event Controller. Whenever a data interrupt is generated, a data Trigger Master pulse is also driven out, if enabled. Each timer has an individual trigger input line, and each timer can be either started or stopped as a Trigger Slave.

In total, the TIMER module can have up to  $(N + 1)$  interrupt output lines and  $N$  data trigger lines.

## TIMER External Interface

Each GP TIMER module can support up to 16 individual timers. However, most processors have less than this number. The exact number of timers available on a given processor is available in that processor's data sheet.

Every timer has one main input/output signal ( $TMR_x$ ) and, usually, one auxiliary input pin, used as an alternate capture input ( $TM\_ACIx$ ). Each TMR can either run with a time base of  $SCLK$  or can reference an external clock on one of two  $TMR\_ALT\_CLKx$  pins. The  $TMR\_ALT\_CLK0$  signal maps to individual alternate clock ( $TM\_ACLKx$ ) pins for one or more timers. For instance, a  $TM\_ACLK3$  pin would provide an alternate site to supply an external signal that would serve as TMR3's reference clock. Likewise, the  $TMR\_ALT\_CLK1$  signal from each timer unit is connected together internally to provide a single global timer clock pin ( $TM0\_CLK$ ) for the GP timer module, for use as an additional time base.

When clocked internally from  $SCLK$ , the maximum period for the timer count is  $((2^{32})-1) / SCLK$  (in MHz)). For  $SCLK=133$  MHz, this would equate to 32.2 seconds. The  $TM\_ACLK$  and  $TM\_ACI$  capture input pins are sampled every  $SCLK$  cycle. The duration of every low or high state must be slightly more than one  $SCLK$  cycle. Therefore the maximum allowed frequency of timer input signals is slightly less than  $SCLK/2$ . For exact timing requirements, please refer to the processor's data sheet).

## TIMER General Operation

The core of every timer is a 32-bit counter that can be interrogated through the read-only  $TIMER\_TMR\_CNT$  register. Once a timer has been enabled, its  $TIMER\_TMR\_CNT$  register is loaded with a starting value.

A timer can operate in one of several different modes, configured through the  $TIMER\_TMR\_CFG$  register for that timer. These modes are known as PWMOUT, EXTCLK, WIDCAP, WATCHDOG, PININT and IDLE, and are summarized in the following table.

**Table 15-5:** Timer Mode Descriptions

| Timer Mode | Description                                                                                                                                                                              |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PWMOUT     | Generates single or continuous PWM waveforms with programmable pulse width, period and delay                                                                                             |
| EXTCLK     | Counts “clock ticks” from the system clock (SCLK) or an externally applied waveform                                                                                                      |
| WIDCAP     | Captures pulse width or period of an externally applied waveform                                                                                                                         |
| WATCHDOG   | Monitors pulse width or period of an external signal and compares against a window of acceptable values, optionally generating an interrupt if it falls inside or outside of that window |
| PININT     | Can generate an interrupt on an active edge applied to a timer pin.                                                                                                                      |
| IDLE       | Idle; no activity                                                                                                                                                                        |

## Timer Period, Width and Delay Register Interaction

When the timer is started, writes to the buffer registers are immediately copied through to the period, pulsewidth, and delay registers. Therefore, these values are ready for use in the first timer period. When a timer is already running, software can write new values to the `TIMER_TMR_PER`, `TIMER_TMR_WID` and `TIMER_TMR_DLY` registers. The written values are buffered and do not update into the registers until the end of the current period (when the value in `TIMER_TMR_CNT` equals the value in `TIMER_TMR_PER`).

If new values are not written to these registers, the value from the previous period is re-used. Writes to these registers are atomic; it is not possible for the high word to be written without the low word also being written. All three registers are double buffered. Values written to the period, pulsewidth, and delay registers are always stored in the buffer registers. Reads from the same register always return the current, active value of period, pulse width or delay value. Written values are not read back until they become active.

The usage of the `TIMER_TMR_PER`, `TIMER_TMR_WID` and `TIMER_TMR_DLY` registers varies, depending on the mode of the timer and is specified by `TIMER_TMR_CFG.TMODE` bits. See the table below for more information.

**Table 15-6:** Usage of the Period, Width and Delay Registers in Different Timer Modes

| Timer Mode | Period       | Width        | DELAY        |
|------------|--------------|--------------|--------------|
| IDLE       | Not writable | Not writable | Not writable |

**Table 15-6:** Usage of the Period, Width and Delay Registers in Different Timer Modes (Continued)

| Timer Mode | Period                                                                                                                                                                                  | Width                                                                                                                                                                                   | DELAY                                                                                                                                                                                   |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WATCHDOG   | Can be updated on-the-fly. New value takes effect either upon timer start or when an asserting edge on the input signal is sensed.                                                      | Read-only. Retains value of last measured width or period of the input signal.                                                                                                          | Can be updated on-the-fly. New value takes effect either upon timer start or when an asserting edge on the input signal is sensed.                                                      |
| WIDCAP     | Read-only. Period value captured at the appropriate time and updated from its buffer register simultaneously with the Width register.                                                   | Read-only. Width value captured at the appropriate time and updated from its buffer register simultaneously with the Period register.                                                   | Not used                                                                                                                                                                                |
| PWMOUT     | Can be updated on-the-fly. New value takes effect either upon timer start or at the end of the current period. A write followed by immediate read returns the current operating values. | Can be updated on-the-fly. New value takes effect either upon timer start or at the end of the current period. A write followed by immediate read returns the current operating values. | Can be updated on-the-fly. New value takes effect either upon timer start or at the end of the current period. A write followed by immediate read returns the current operating values. |
| EXTCLK     | Can be updated on-the-fly.                                                                                                                                                              | Not used                                                                                                                                                                                | Not used                                                                                                                                                                                |
| PININT     | Not used                                                                                                                                                                                | Not used                                                                                                                                                                                | Not used                                                                                                                                                                                |

**NOTE:** It is important to note that, if in a particular timer mode any of these three registers is not used, then software is not allowed to write into the unused one(s). For example, in WIDCAP mode, delay registers are not used. Therefore, software is not allowed to write any value to `TIMER_TMR_DLY`. The software must program `TIMER_TMR_CFG.TMODE` bits before programming these 3 registers, in order to prevent undesired operation.

Also, if software is changing `TIMER_TMR_CFG.TMODE` bits such that these registers change configuration from status register to writable register (for example for PWMOUT mode), hardware does not clear these registers. These values are automatically overwritten by new values specified by software.

In PWM\_OUT mode with very small periods, there may not be enough time between updates from the buffer registers to write these registers; the next period may use one old value and one new value. In order to prevent  $(\text{width} + \text{pulse delay}) > \text{period}$  errors, write the width and delay registers before the period register when decreasing the values, and write the period register before the width and delay registers when increasing the value.

## TIMER Programming Concepts

Using the features, operating modes, and event control for the GP Timer to their greatest potential requires an understanding of some GP Timer related concepts.

### Setting Up Constantly Changing Timer Conditions

This task shows how to use different period, pulse width, and/or delay settings for each of the first three timer periods after the timer is started.

1. Program the first set of period, width and delay register values.
2. Enable the timer.
3. Immediately program the second set of register values, as needed.
4. Wait for the first timer interrupt.
5. Program the third set of register values.

*RESULT:*

Each new setting is then programmed when the preceding timer interrupt is received.

### Configuring, Enabling and Disabling One or More Timers

1. Configure the relevant timer(s) for the operating mode and other properties through the `TIMER_TMR_CFG` register.
2. Write a 1 to the representative `TIMER_RUN` bit(s) or, alternately, use the `TIMER_RUN_SET` register to avoid disturbing the settings of other timers not being presently configured.

*STEP RESULT:* The timer(s) should now be enabled and operating.

3. To stop one or more timers, first program the `TIMER_STOP_CFG` register to determine whether to stop immediately or gracefully upon receiving a stop command.

*ADDITIONAL INFORMATION:* Note that PWMOUT modes are the only modes where a timer can be configured for graceful termination.

4. Write a 0 to the representative `TIMER_RUN` bit(s) to stop the timer(s) according to their `TIMER_STOP_CFG` settings. Alternately, write a 1 to the appropriate `TIMER_RUN_CLR` bits to avoid disturbing the settings of other timers not being presently stopped.

*STEP RESULT:* The timer(s) should now be stopped.



## Configuring Timer Data and Status Interrupts

1. Program the proper value in the `TIMER_TMR_IRQMODE` field, according to the desired interrupt properties.
2. Unmask the interrupt source at the system event controller.
3. To poll the timer's `TIMER_DATA_ILAT` bit without generating an interrupt, set the `IRQMODE` field but leave the interrupt masked at the system level.
4. If enabled by the `TIMER_STAT_IMSK` register, interrupt requests are also generated by overflow or error conditions (wrong programming values), as reported by the `TMR_STAT_ILAT` bits, provided that the timer status interrupt source is unmasked at the system event controller.
5. To poll the timer's `TIMER_STAT_ILAT` bit without generating an interrupt, the corresponding bit must be unmasked in the `TIMER_STAT_IMASK` register but leave the interrupt masked at the system level.

## Using the Timer Broadcast Feature

The broadcast feature provides a means to update period, width and/or delay registers simultaneously across more than one timer.

1. Enable the appropriate broadcast bits (`BPEREN`, `BWIDEN`, `BDLYEN`) in the `TIMER_TMR_CFG` registers for the timers involved in the broadcast. The broadcast bits use depend on which `TIMER_BCAST` registers are involved.
2. Program the `TIMER_BCAST_PER` register (for instance), assuming you want to broadcast the period setting across the multiple timers enabled above.

*STEP RESULT:* This causes only those timers enabled above to load their `TIMER_TMR_PER` registers with the value specified in the `TIMER_BCAST_PER` register.

3. Repeat Step 2 as needed for `TIMER_BCAST_WID` and `TIMER_BCAST_DLY` settings.

## Timer Single-Pulse PWMOUT Mode

In single-pulse PWMOUT mode, the timer generates a single pulse on the TMR pin. This mode is frequently used to implement a precise delay, often in conjunction with generation of an output trigger. The assertion of a pulse is controlled by the value in `TIMER_TMR_DLY`, and pulse width is defined by the `TIMER_TMR_WID` value. `TIMER_TMR_PER` is not used and cannot be written in this mode. After completion of the pulse the timer is stopped automatically, optionally generating an interrupt, if configured to do so. Pulse polarity is controlled through the `TIMER_TMR_CFG.PULSEHI` bit.

If configured as such, the timer can generate a data interrupt upon satisfying various conditions specified by the `TIMER_TMR_CFG.IRQMODE` bits.

It is not necessary to clear the relevant `TIMER_RUN` bit in order to stop the timer cleanly. At the end of the pulse, the timer stops automatically and the corresponding `TIMER_RUN` is cleared. To generate multiple discrete pulses (as opposed to a continuous PWM waveform), write a 1 to the appropriate `TIMER_RUN` bit, wait for the timer to stop, and then write another 1 to the same `TIMER_RUN` bit.

## Timer Continuous PWMOUT Mode

In continuous PWMOUT mode, the timer generates repetitive pulses with well-defined period, duty cycle and pulse position. The `TIMER_TMR_DLY`, `TIMER_TMR_PER` and `TIMER_TMR_WID` registers are programmed with the values of the required PWM pulse. After the timer is started, the counter counts towards the value programmed in `TIMER_TMR_PER`. Initially, the TMR pin remains in a de-asserted state. It toggles to an asserted state when `TIMER_TMR_CNT=TIMER_TMR_DLY`. The assertion sense of the TMR pin can be controlled with the `TIMER_TMR_CFG.PULSEHI` bit. The TMR pin holds this value for the number of clock cycles specified in `TIMER_TMR_WID`, after which it de-asserts and holds this value until the completion of the programmed period. The same waveform is generated repeatedly until the timer is disabled.

If configured as such, the timer can generate a data interrupt upon satisfying any of various conditions specified by the `TIMER_TMR_CFG.IRQMODE` bits.

It is important to guarantee that the programmed Period  $\geq$  (Width+Delay). Similarly, delay must be less than period. Violating either of these criteria will result in an unpredictable waveform on the TMR pin until the situation is rectified by writing proper values to these registers.

The maximum frequency possible to generate on the TMR pin is achieved by setting `TIMER_TMR_PER` to 2 and `TIMER_TMR_WID` to 1. This makes the TMR pin toggle each `SCLK` clock cycle (assuming the timer is configured to clock internally), producing a duty cycle of 50%. For instance, if `SCLK` is running at 166 MHz, the maximum frequency generated at the TMR pin is  $((2^{32} - 1) / 166 \text{ MHz})$ , or 25.87 seconds.

When a timer's `TIMER_STOP_CFG` bit is 0, the timer treats a stop operation as a stop is pending condition. When terminated with this setting, the timer automatically completes the current waveform and then stops cleanly, remaining in a deasserted state. This prevents truncation of the current pulse and unwanted PWM patterns at the TMR pin. The processor can determine when the timer stops running by polling the corresponding `TIMER_RUN` bit until it reads as read 0 or by waiting for the last interrupt (if enabled).

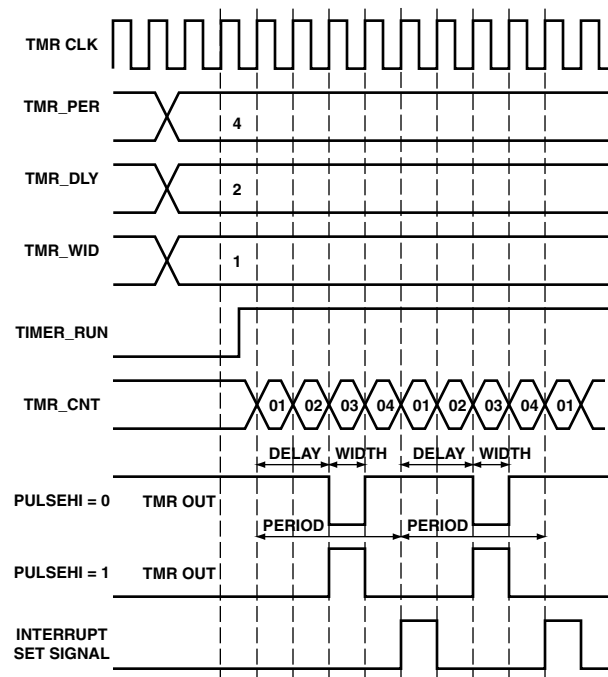


**Figure 15-1:** Signal Generation in Continuous PWMOUT Mode

Note that the `TIMER_TMR_CFG` register cannot be reconfigured until after the timer stops and `TIMER_RUN` reads 0.

If required, the software can force a timer to stop immediately in PWM\_OUT mode by writing a 1 into `TIMER_STOP_CFG` followed by writing a 1 to `TIMER_RUN_CLR` (or by writing a 0 to the appropriate `TIMER_RUN` bit). This stops the timer whether the pending stop was waiting for the end of the current period or the end of the current pulse width. This feature may be used to regain immediate control of a timer during an error recovery sequence.

Use this feature carefully, because it may corrupt the PWM pattern generated at the TMR pin, though after such a stop the pin deasserts automatically. Each timer samples its `TIMER_RUN` bit at the end of each period. It stops cleanly at the end of the first period after `TIMER_RUN` is low. This implies that a timer that is disabled and then re-started, all before the end of the current period, will continue to run as if nothing happened. Typically, software should disable a PWMOUT timer and then wait for it to stop itself.

## TIMER Width Capture (WIDCAP) Mode

The WIDCAP mode, often simply called capture mode, is used to measure pulse widths on the TMR or TMR\_AUX\_IN inputs. The polarity (active high/low) of the input signal can be selected with the `TIMER_TMR_CFG.PULSEHI` bit. The figure below shows the control signal flow for WIDCAP\_CAP mode.

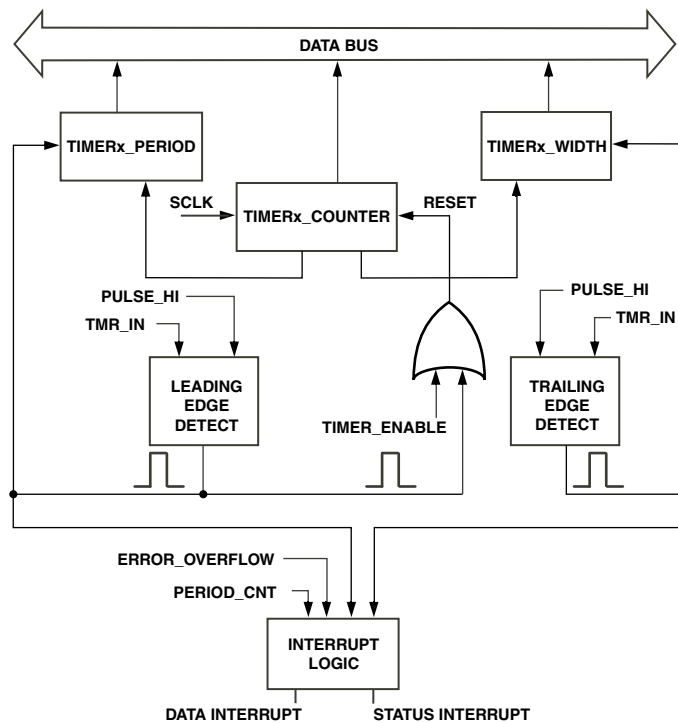


**Figure 15-2:** Timer Signal Flow in Width Capture Mode

In this mode, the `TIMER_TMR_CFG.TINSEL` bit selects between the TMR or TMR\_AUX\_IN input. The internally clocked timer is used to determine the period and pulse width of the externally applied rectangular waveforms.

When a timer is enabled in this mode, the timer resets the count in its `TIMER_TMR_CNT` register to 0x0000 0001 and does not start counting until it detects a leading edge on the selected input pin.

When the timer detects the first leading edge, it starts incrementing. When it detects a trailing edge of a waveform, it captures the current 32-bit value of its `TMR_CNT` register into its width buffer register. At the next leading edge, the timer transfers the current 32-bit value of its `TMR_CNT` register into its period buffer register. The `TMR_CNT` register is reset to 0x0000 0001 again, and the timer continues counting and capturing until it is disabled.

In this mode, software can measure both the pulse width and the pulse period of a waveform. The `TMR_DLY` register is not used in this mode. The `TIMER_TMR_CFG.PULSEHI` bit controls the definition of leading edge and trailing edge of the TMR/TMR\_AUX\_IN pin.

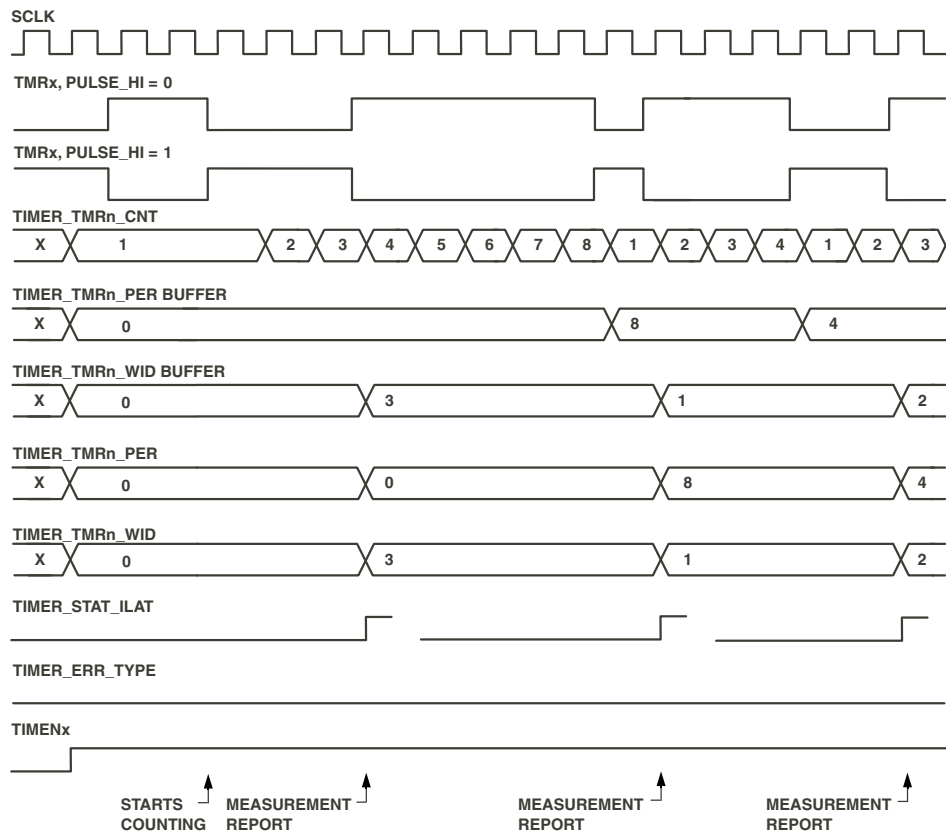
In WIDCAP mode, the following events always occur at the same time as one unit:

1. The `TIMER_TMR_PER` register is updated from the period buffer register.
2. The `TIMER_TMR_WIDTH` register is updated from the width buffer register.
3. The `TIMER_DATA_ILAT` bit gets set (if enabled).

4. A timer data trigger pulse is generated (if enabled).

The `TIMER_TMR_CFG.TMODE[0]` bit controls the point in time at which this set of events is executed. Taken together, these four events are called a measurement report. The `TMR_STAT_ILAT` register does not get set at a measurement report. A measurement report occurs, at most, once per input signal period. The current `TMR_CNT` value is always copied to the width buffer and period buffer registers at the trailing and leading edges of the input signal, respectively, but these values are not visible to software. A measurement report event samples the captured values into visible registers and sets the timer interrupt to signal that `TMR_PER` and `TMR_WID` are ready to be read.

When `TMODE=b#1011`, the measurement report occurs just after the width buffer register captures its value (at a falling edge). Subsequently, the `TMR_WID` register reports the pulse width measured in the pulse that has just ended, but the `TMR_PER` register reports the pulse period measured at the end of the previous period. This is because, if only the first trailing edge occurred, then the first period value has not yet been measured at the first measurement report, so the period value is not valid. A read of the `TMR_PER` value in this case returns 0. See the following figure for more information.



NOTE: FOR SIMPLICITY, THE SYNCHRONIZATION DELAY BETWEEN TMRx EDGES AND BUFFER REGISTER UPDATES IS NOT SHOWN.

**Figure 15-3:** Example of Width Capture Deasserted Mode (`TMODE=b#1011`)

When  $TMODE=b\#1010$ , the measurement report occurs just after the period buffer register captures its value (at a leading edge). Subsequently, the  $TMR\_PER$  and  $TMR\_WID$  registers report the pulse period and pulse width is measured in the period that has just ended. Refer to the following figure for more information.

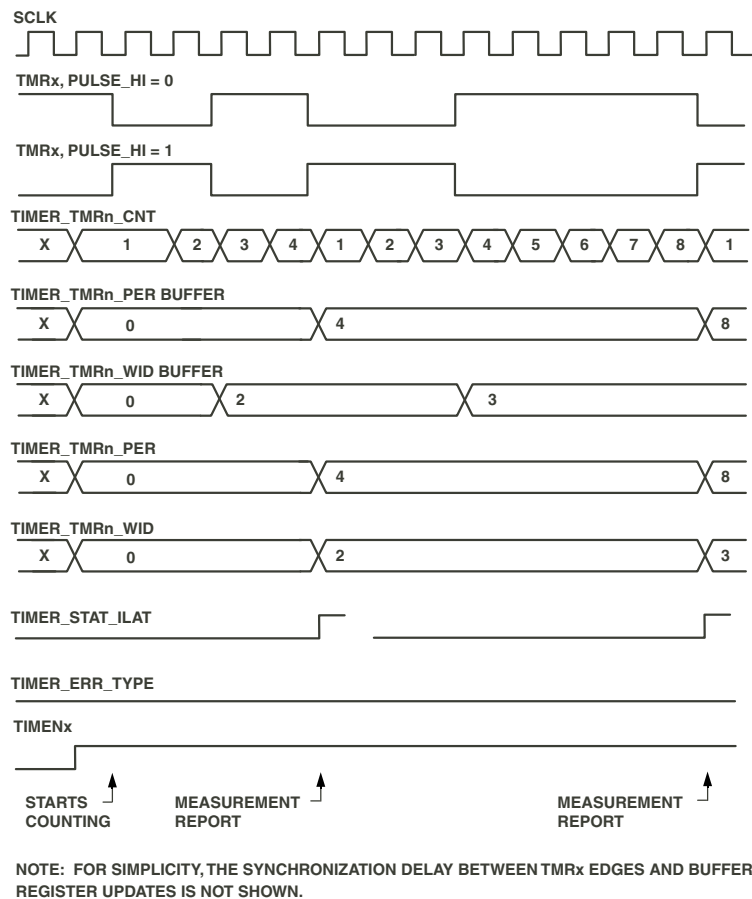


**Figure 15-4:** Example of Width Capture Asserted Mode ( $TMODE=b\#1010$ )

To measure the pulse width of a waveform that has only one leading edge and one trailing edge, set  $TMODE = b\#1011$ . If  $TMODE = b\#1010$  for this case, no period value is captured in the period buffer register. Instead, an error report interrupt is generated (if enabled) when the  $TMR\_CNT$  range is exceeded and the counter wraps around. In this case, both  $TMR\_WID$  and  $TMR\_PER$  read 0 (because no measurement report occurred to copy the value captured in the width buffer register to  $TMR\_WID$ ).

If using the  $TMODE = b\#1010$  mode to measure the width of a single pulse, it is recommended to disable the timer after taking the interrupt that ends the measurement interval. If desired, the timer can then be restarted as appropriate in preparation for another measurement. This procedure prevents the timer from free-running after the width measurement and logging errors generated by the timer count overflowing.

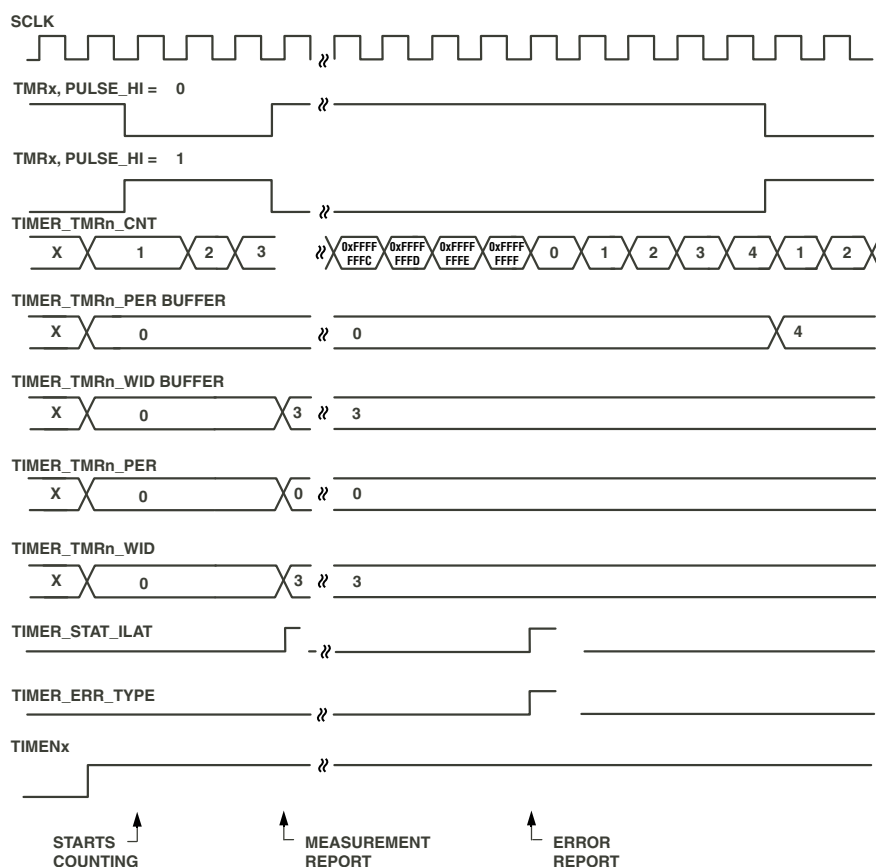
## GP Timer Width Capture Mode Overflow

A timer status interrupt (if enabled) is generated if the `TMR_CNT` register wraps around from `0xFFFF FFFF` to 0 in the absence of a leading edge. At that point, the timer's `TIMER_STAT_ILAT` bit gets set and the `TIMER_ERR_TYP` bits change appropriately, indicating a count overflow due to a period greater than the counter's range. This is called an error report. A data interrupt in WIDCAP mode indicates a new measurement is ready to be read (a measurement report). Similarly, an interrupt on the timer status interrupt line (shared interrupt for all timers) indicates an overflow if generated in WIDCAP mode.

The `TMR_PER` and `TMR_WID` registers are never updated at the time an overflow is signaled. If the timer overflowed and `TMODE=b#1010`, neither the `TMR_PER` nor the `TMR_WID` register were updated. If the timer overflowed and `TMODE=b#1011`, the `TMR_PER` and `TMR_WID` registers were updated only if a trailing edge was detected at a previous measurement report.

Software can count the number of error reports between measurement report interrupts to measure input signal periods longer than `0xFFFF FFFF`. Each error report interrupt adds a full  $2^{32}$  SCLK counts to the total for the period, but the width is ambiguous. Make sure that, if only the status interrupt is monitored by software in this case, then status interrupts from all other timers are masked.

For example, in the following figure, the period is `0x1 0000 0004`, but the pulse width could be either `0x0 0000 0002` or `0x1 0000 0002`.



NOTE: FOR SIMPLICITY, THE SYNCHRONIZATION DELAY BETWEEN TMRx EDGES AND BUFFER REGISTER UPDATES IS NOT SHOWN.

**Figure 15-5:** Example Timing for Width Capture Followed by Period Overflow (TMR\_CFG.TMODE=b#1010)

The waveform applied to the TMR (or TMR\_AUX\_IN) pin is not required to have a 50% duty cycle, but the minimum input low time is little more than one SCLK period and the minimum input high time is little more than one SCLK period (refer to the product data sheet for details). This implies the maximum TMR input frequency is somewhat less than  $SCLK/2$ , with a 50% duty cycle. Under these conditions, the WIDCAP mode timer would measure Period = 2 and Pulse Width = 1.



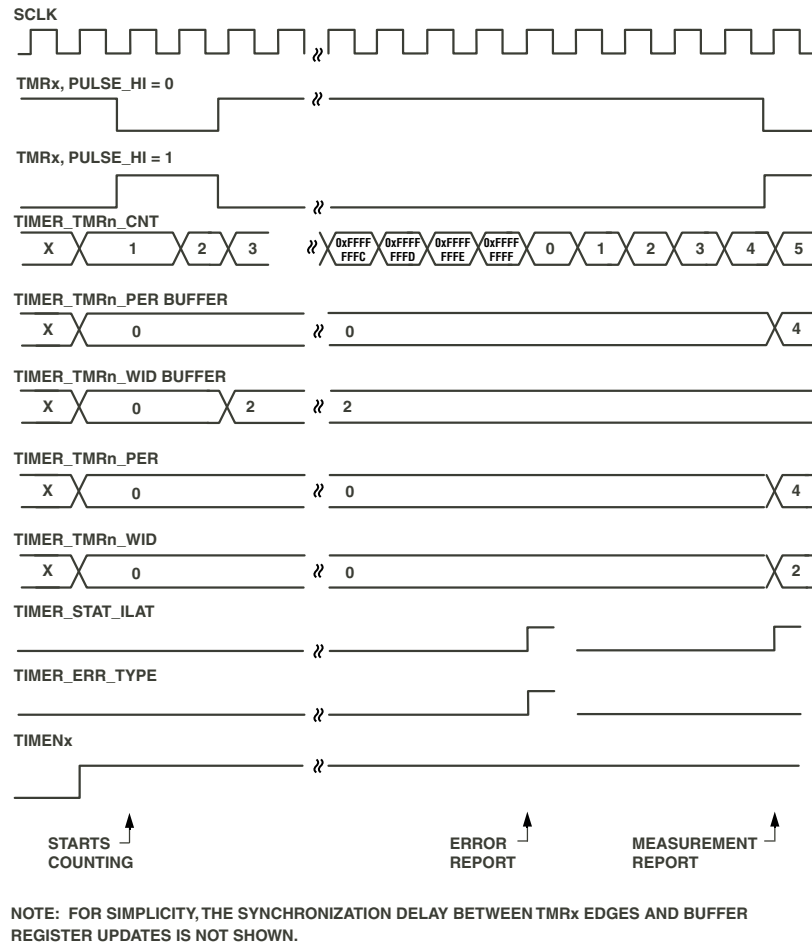


**Figure 15-6:** Example Timing for Width Capture Followed by Period Overflow (TMR\_CFG.TMODE=b#1011)

## TIMER Windowed Watchdog (WATCHDOG) Modes

In windowed watchdog (WATCHDOG) modes, a timer can take inputs from either the TMR pin or the TMR\_AUX\_IN pin. With this mode, the timer can monitor pulse width (width watchdog mode) or pulse period (period watchdog mode) on the input line. It also compares the measured value against a minimum required value and maximum allowed value and generates an interrupt appropriately. Polarity selection of the input signal is performed by the `TIMER_TMR_CFG.PULSEHI` bit.

The waveform applied to the input pin in watchdog mode is not required to have a 50% duty cycle, but the minimum input pulse low time is slightly more than one SCLK period, and the minimum input pulse high time is slightly more than one SCLK period (refer to the product data sheet for details). This implies the maximum input frequency is somewhat less than  $SCLK/2$  in this mode.

## Timer Windowed Watchdog Width Mode

In this mode, the timer counter monitors the pulse width of an input signal on either the TMR pin or TMR\_ALT\_CLK pin. Software needs to program the minimum pulse width (Pmin) in the `TIMER_TMR_DLY` register and the maximum pulse width (Pmax) in the `TIMER_TMR_PER` register. Both values are programmed in terms of number of clock cycles (SCLK or TMR\_ALT\_CLKx). The timer can generate an interrupt if the deasserting pulse edge occurs inside the window ( $P_{min} < \text{Pulse Width} \leq P_{max}$ ) or outside the window ( $\text{Pulse Width} \leq P_{min}$  or  $\text{Pulse Width} > P_{max}$ ).

After enabling the timer in this mode, it always starts counting at the asserting edge of the input signal. This means any pulse that is already active when the timer is enabled is ignored.

With `TIMER_TMR_CFG.IRQMODE=b#11`, the timer generates an interrupt (if enabled) if the timed pulse width exceeds Pmax, or if the pulse width is less than Pmin. After attaining Pmax, the pulse still remains at an active level, and the counter keeps on counting until it sees a deasserting edge. When the input pulse is not active, the counter holds its current value until it again sees an asserting edge, or it restarts. An interrupt can also be generated for when the pulse occurs within the specified window condition, by setting `TIMER_TMR_CFG.IRQMODE=b#10`.

In this mode, a trailing edge on the input pin triggers capturing of pulse width into the `TIMER_TMR_WID` register. During the inactive portion of the input signal, the internal counter does not increment. Refer to the figure below for signal flow in this mode.

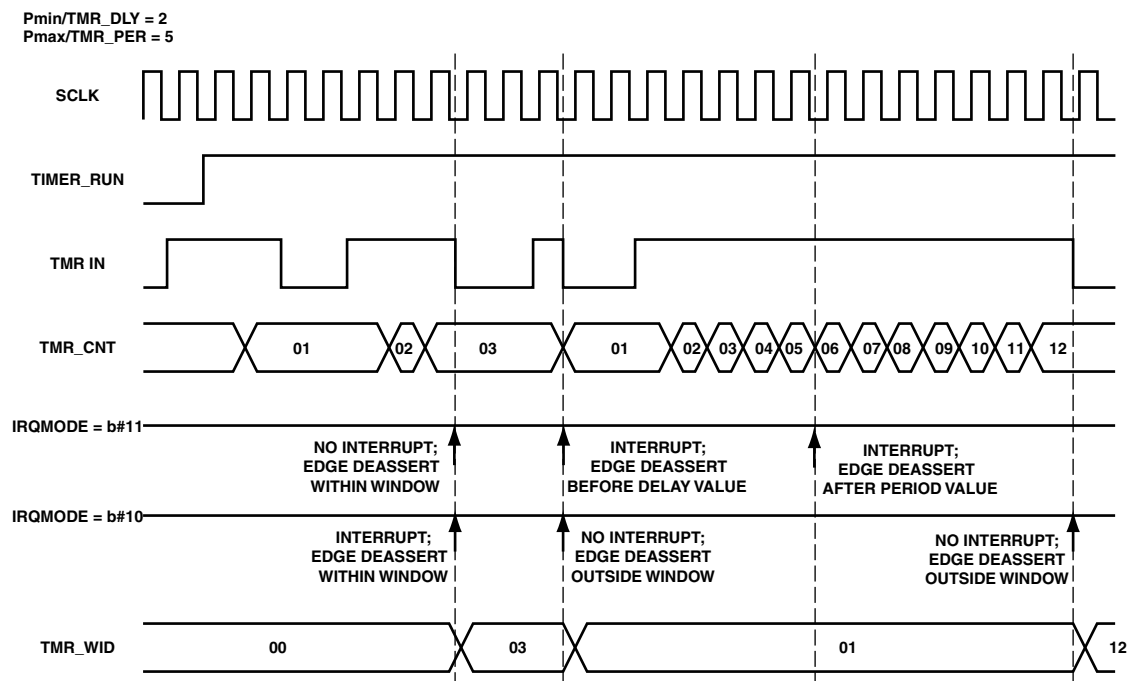


**Figure 15-7:** Watchdog Width Mode Timing

If it is required to check only the upper limit on pulse width (Pmax but not Pmin) then Pmin must be programmed as 0 or 1. In such a case, it is better to use `IRQ_MODE = b#11`. With `IRQ_MODE = b#10`, a pulse width of 1 clock cycle will result in an interrupt. For details see the table below.

**Table 15-7:** Windowed Watchdog Width Mode Interpretation

| Timer Delay                | Timer Period      | Incoming Pulse Width        | IRQMODE= b#10                                 | IRQMODE= b#11                                                  | Error Interrupt?   |
|----------------------------|-------------------|-----------------------------|-----------------------------------------------|----------------------------------------------------------------|--------------------|
| 0 or 1                     | Anything $\geq 1$ | PW = 1                      | Interrupt at deasserting edge of input signal | No Interrupt                                                   | No Error Interrupt |
|                            |                   | PW $\leq$ TMR_PER           | Interrupt at deasserting edge of input signal | No Interrupt                                                   | No Error Interrupt |
|                            |                   | PW > TMR_PER                | No Interrupt                                  | Interrupt when Pulse with exceeds Pmax (Period Register) Value | No Error Interrupt |
| > 1 but $\leq$ (Period -1) | Anything > 1      | PW $\leq$ TMR_DLY           | No Interrupt                                  | Interrupt at De-asserting edge of input Signal                 | No Error Interrupt |
|                            |                   | TMR_DLY < PW $\leq$ TMR_PER | Interrupt at deasserting edge of input signal | No Interrupt                                                   | No Error Interrupt |
|                            |                   | PW > TMR_PER                | No Interrupt                                  | Interrupt when Pulse with exceeds Pmax (Period Register) Value | No Error Interrupt |
| $\geq$ Period              | -                 | PW $\leq$ TMR_PER           | Undefined                                     | Undefined                                                      | No Error Interrupt |
|                            | -                 | PW > TMR_PER                | Undefined                                     | Undefined                                                      | b#11 Error Type    |
| -                          | 0                 | -                           | Undefined                                     | Undefined                                                      | b#10 Error Type    |

Timer Windowed Watchdog Period Mode

In this mode, the timer monitors the number of clock cycles between two consecutive rising/falling edges of an input signal on either the TMR pin or TMR\_AUX\_IN pin. Software needs to program the required minimum number of clock cycles (Tmin) in the TIMER\_TMR\_DLY register and the required maximum allowed number of clock cycles (Tmax) in the TIMER\_TMR\_PER register. Both values are programmed in terms of number of clock cycles (SCLK or TMR\_ALT\_CLKx). The timer can generate an interrupt if two consecutive occurrences of an active edge are within a specified window ( $T_{min} < \text{Pulse Period} \leq T_{max}$ ) or outside ( $\text{Pulse Width} \leq T_{min}$  or  $T_{max} < \text{Pulse Width}$ ) a specified window.

With `TIMER_TMR_CFG.IRQMODE=b#11`, if the pulse period ever exceeds Tmax or if it is less than or equal to Tmin, the timer generates an interrupt if unmasked. After attaining the Tmax value, the counter keeps on counting until it sees an active edge on the input line. An interrupt can also be generated for when the pulse occurs within the specified window condition, by setting `TIMER_TMR_CFG.IRQMODE=b#10`. Refer to the figure below for timer functionality in period watchdog mode.

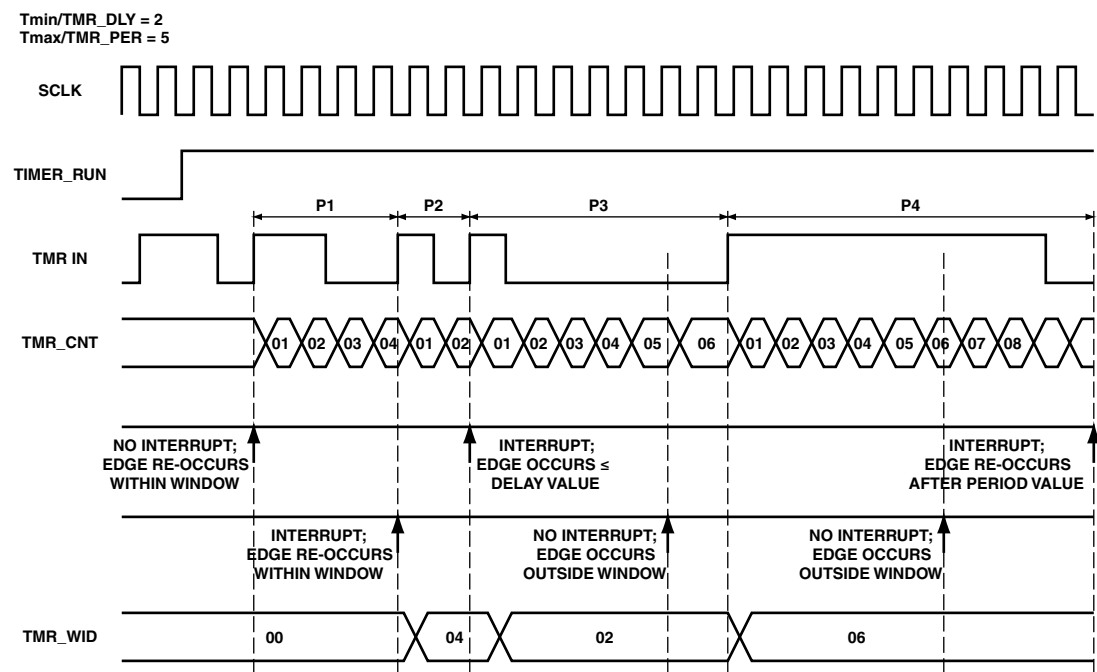


Figure 15-8: Watchdog Period Mode Timing

If it is required to check only the upper limit on period (Tmax value but not Tmin value) then Tmin can be programmed as 0 or 1 in this mode. For details refer to the table below.

**Table 15-8:** Windowed Watchdog Period Mode Interpretation

| Timer Delay                   | Timer Period      | Incoming Pulse Width                    | IRQMODE=b#10                                  | IRQMODE =b#11                                                    | Error Interrupt?   |
|-------------------------------|-------------------|-----------------------------------------|-----------------------------------------------|------------------------------------------------------------------|--------------------|
| 0 or 1                        | Anything $\geq 2$ | Pulse Period $\leq$ TMR_PER             | Interrupt at deasserting edge of input signal | No Interrupt                                                     | No Error Interrupt |
|                               |                   | Pulse Period $>$ TMR_PER                | No Interrupt                                  | Interrupt when pulse period crosses Pmax (Period Register) value | No Error Interrupt |
| $\geq 1$ but $\leq$ Period -1 | Anything $\geq 2$ | Pulse Period $\leq$ TMR_DLY             | No Interrupt                                  | Interrupt at deasserting edge of input signal                    | No Error Interrupt |
|                               |                   | TMR_DLY $<$ Pulse Period $\leq$ TMR_PER | Interrupt at deasserting edge of input signal | No Interrupt                                                     | No Error Interrupt |
|                               |                   | Pulse Period $>$ TMR_PER                | No Interrupt                                  | Interrupt when pulse width exceeds Pmax (Period Register) value  | No Error Interrupt |
| $\geq$ Period                 | -                 | Pulse Period $<$ TMR_PER                | Undefined                                     | Undefined                                                        | No Error Interrupt |
|                               |                   | Pulse Period $\geq$ TMR_PER             | Undefined                                     | Undefined                                                        | b#11 Error Type    |
| -                             | 0 or 1            | -                                       | Undefined                                     | Undefined                                                        | b#10 Error Type    |

## TIMER Pin Interrupt (PININT) Mode

In PININT mode, any active edges on either the TMR pin or the TMR\_AUX\_IN pin (whichever is selected by `TIMER_TMR_CFG.TINSEL`) can cause an edge-based interrupt if enabled. The event on the input pin can

set the `TIMER_DATA_ILAT` bit and issue a system interrupt request. Active edge polarity can be changed by programming the `TIMER_TMR_CFG.PULSEHI` bit.

Since the interrupt is generated in the `SCLK` clock domain, the width of the input signal must be more than one `SCLK` period. Along with generating the interrupt, the timer will also generate a trigger pulse if it is enabled in the `TIMER_TRG_MSK` register. Due to configuration of polarity, glitches at the input may cause an undesired interrupt to be generated. To avoid this, software must ensure that interrupts are unmasked only after configuring desired polarity.

## TIMER External Clock (EXTCLK) Mode

Use the `EXTCLK` mode, sometimes referred to as the counter mode, to count external events, that is, signal edges on either the `TMR` or `TMR_AUX_IN` input pin. The timer works as a counter clocked by an external source (the signal at the pin), which can be asynchronous to `SCLK`. The current count in `TIMER_TMR_CNT` represents the number of leading edge events detected. The `TIMER_TMR_PER` register is programmed with the value of the maximum timer external count desired before stopping and/or issuing an interrupt or trigger.

The `TIMER_TMR_CFG.PULSEHI` bit determines the polarity of the leading edge on the input pin. The `TIMER_TMR_CFG.TINSEL` bit selects whether event is counted on `TMR` pin or on the `TMR_AUX_IN` pin. The `TIMER_STAT_ILAT` and `TIMER_ERR_TYP` bits are set if `TIMER_TMR_CNT` wraps around from `0xFFFF FFFF` to 0 or if the period = 0 at startup or when `TIMER_TMR_CNT` rolls over (from count = period to count = 0x1). The `TIMER_TMR_WID` and `TIMER_TMR_DLY` registers are unused in this mode.

The figure below shows a flow diagram for `EXTCLK` mode.

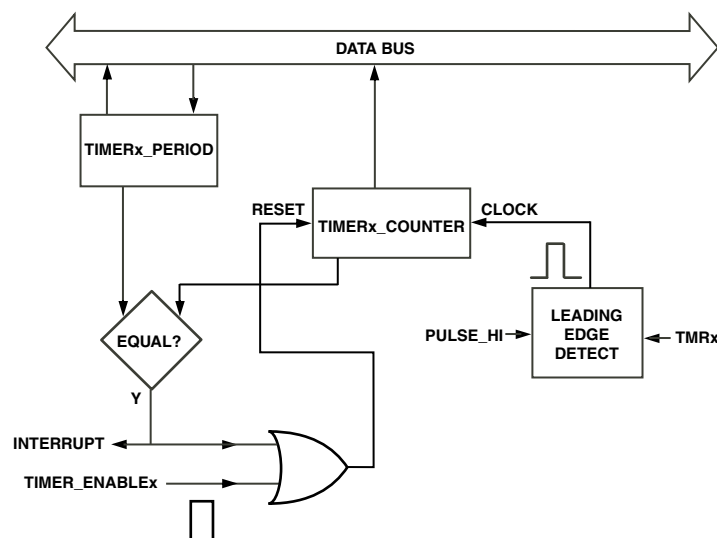


**Figure 15-9:** EXTCLK Mode Control Flow

The waveform applied to the input pin is not required to have a 50% duty cycle, but the minimum input low time and input high time are both slightly more than one `SCLK` period, (refer to the product data sheet

for details). This implies the maximum input frequency is slightly less than  $SCLK/2$ . The period may be programmed to any value from 1 to  $(2^{32} - 1)$ , inclusive.

After the timer has started, it resets `TIMER_TMR_CNT` to 0x0 and then waits for the first leading edge on the input pin. This edge causes `TMR_CNT` to be incremented to the value 0x1, and every subsequent leading edge increments it by one. After `TMR_CNT` reaches the value programmed in `TIMER_TMR_PER`, the corresponding `TIMER_DATA_ILAT` bit is set, and an interrupt and trigger are both generated (if enabled). The next leading edge reloads the `TMR_CNT` again with 0x1, and the timer continues counting until it is disabled.

## Timer Illegal States

These definitions are used in the table below:

- **Startup.** The first clock period during which the timer counter is running after the timer is started by writing the `TIMER_RUN` register.
- **Rollover.** The time when the current count in `TMR_CNT` matches the value in `TMR_PER` and the counter is reloaded with the value 1.
- **Overflow.** The timer counter was incremented instead of doing a rollover when it was holding the maximum possible count value of 0xFFFF FFFF. The counter does not have a large enough range to express the next greater value and so erroneously loads a new value of 0x0000 0000.
- **Unchanged.** No new error.

When `ERR_TYPE` is designated unchanged, it displays the previously reported error code orb# 00 if there has been no error since this timer was enabled.

When `TIMER_STAT_ILAT` is unchanged, it reads 0 if there has been no error or overflow since this timer was enabled, or if software has performed a W1C to clear any previous error. If a previous error has not been acknowledged by software, `STAT_ILAT` reads 1. Software should read `STAT_ILAT` to check for an error. If a particular timer's bit is set there, software can then read `TIMER_ERR_TYPE` for more information. Once detected, software should W1C the appropriate `STAT_ILAT` bit to acknowledge the error.

The following tables can be read as: “In mode \_\_ at event \_\_, if `TMR_PER` is \_\_ and `TMR_WID` is \_\_ and `TMR_DLY` is \_\_, then `TIMER_ERR_TYPE` is \_\_ and `TIMER_STAT_ILAT` is \_\_.” Startup error conditions do not prevent the timer from starting. Similarly, overflow and rollover error conditions do not stop the timer. Illegal cases may cause unwanted behavior of the TMR pin.

**NOTE:** For PININT mode error functionality is not used.

## Continuous PWMOUT Mode

**Table 15-9:** Startup Event

| TMR_PER | TMR_DLY                                             | MR_WID                  | TMR_WID +<br>TMR_DLY | ERR_TYPE                                  | TIMER_STAT_ILAT (if<br>enabled)     |
|---------|-----------------------------------------------------|-------------------------|----------------------|-------------------------------------------|-------------------------------------|
| =< 1    | Anything other<br>than period[8]                    | Anything                | Anything             | b#10                                      | Set                                 |
| >= 2    | Anything including<br>0, excluding<br>TMR_PER value | Anything<br>including 0 | =< PERIOD            | Unchanged                                 | Unchanged                           |
|         | Anything including<br>0                             | Anything<br>including 0 | > PERIOD             | Unchanged[9]<br>(Detected at<br>rollover) | Unchanged (Detected<br>at rollover) |
|         | Anything                                            | Anything                | $> 2^{32} - 1$       | b#11                                      | Set                                 |
|         | =Period                                             | =0                      | =Period              | No error                                  | Unchanged (Detected<br>at rollover) |

**Table 15-10:** Rollover Event

| TMR_PER | TMR_DLY  | TMR_WID  | TMR_WID +<br>TMR_DLY | ERR_TYPE                                                                 | TIMER_STAT_IL<br>AT (if enabled) |
|---------|----------|----------|----------------------|--------------------------------------------------------------------------|----------------------------------|
| =<1     | Anything | Anything | Anything             | b#10[timer<br>running at SCLK]<br>b#11 [timer<br>running at<br>ALT_CLKx] | Set                              |



**Table 15-10:** Rollover Event (Continued)

| TMR_PER | TMR_DLY                                       | TMR_WID              | TMR_WID +<br>TMR_DLY | ERR_TYPE  | TIMER_STAT_IL<br>AT (if enabled) |
|---------|-----------------------------------------------|----------------------|----------------------|-----------|----------------------------------|
| >= 2    | Anything including 0, excluding TMR_PER value | Anything including 0 | =<PERIOD             | Unchanged | Unchanged                        |
|         | Anything including 0, excluding TMR_PER value | Anything >0          | >PERIOD              | b#11      | Set                              |
|         | Anything                                      | Anything             | $> 2^{32} - 1$       | b#11      | Set                              |
|         | = Period[10]                                  | =0                   | =Period              | b#11      | Set                              |
|         | >Period                                       | =0                   | >Period              | Unchanged | Unchanged                        |

**Table 15-11:** Overflow Event (On TMR\_PER Register Programming Error Only)

| TMR_PER  | TMR_DLY  | TMR_WID  | TMR_WID +<br>TMR_DLY | ERR_TYPE | TIMER_STAT_ILAT (if<br>enabled) |
|----------|----------|----------|----------------------|----------|---------------------------------|
| Anything | Anything | Anything | Anything             | b#01     | Set                             |

## Single Pulse PWMOUT Mode

For Single Pulse PWMOUT mode, there are no rollover events.

**Table 15-12:** Startup Event

| TMR_PER | TMR_DLY              | TMR_WID | TMR_WID +<br>TMR_DLY | ERR_TYPE<br>TIMER_STAT<br>_ILAT (if<br>enabled) |           |
|---------|----------------------|---------|----------------------|-------------------------------------------------|-----------|
| NA      | Anything             | == 0    | Anything             | b#11[11]                                        | Set       |
| NA      | Anything including 0 | >=1     | $> 2^{32} - 1$       | Unchanged                                       | Unchanged |
| NA      | Anything including 0 | >=1     | $> 2^{32} - 1$       | b#11                                            | Set       |

**Table 15-13:** Overflow Event (On another error, such as DELAY + WIDTH  $\geq 2^{32} - 1$ )

| TMR_PER  | TMR_DLY  | TMR_WID  | TMR_WID +<br>TMR_DLY | ERR_TYPE | TIMER_STA_ILAT (if<br>enabled) |
|----------|----------|----------|----------------------|----------|--------------------------------|
| Anything | Anything | Anything | Anything             | b#01     | Set                            |

## WID CAP Mode

For WID CAP mode, the TMR\_PER and TMR\_WID registers are read-only and the TMR\_DLY register is not used. Therefore no startup or rollover errors are possible.

**Table 15-14:** Overflow Event

| TMR_PER  | TMR_DLY | TMR_WID  | TMR_WID +<br>TMR_DLY | ERR_TYPE | TIMER_STAT_ILAT (if<br>enabled) |
|----------|---------|----------|----------------------|----------|---------------------------------|
| Anything | NA      | Anything | NA                   | b#01     | Set                             |

## EXTCLK Mode

**Table 15-15:** Startup Event

| TMR_PER  | TMR_DLY | TMR_WID | TMR_WID +<br>TMR_DLY | ERR_TYPE  | TIMER_STAT_ILAT (if<br>enabled) |
|----------|---------|---------|----------------------|-----------|---------------------------------|
| =0       | NA      | NA      | NA                   | b#01      | Set                             |
| $\geq 1$ | NA      | NA      | NA                   | Unchanged | Unchanged                       |

**Table 15-16:** Rollover Event

| TMR_PER  | TMR_DLY | TMR_WID | TMR_WID +<br>TMR_DLY | ERR_TYPE  | TIMER_STAT_ILAT (if<br>enabled) |
|----------|---------|---------|----------------------|-----------|---------------------------------|
| =0       | NA      | NA      | NA                   | b#01      | Set                             |
| $\geq 1$ | NA      | NA      | NA                   | Unchanged | Unchanged                       |

**Table 15-17:** Overflow Event (On TMR\_PER Register = 0 Only)

| TMR_PER  | TMR_DLY | TMR_WID | TMR_WID +<br>TMR_DLY | ERR_TYPE | TIMER_STAT_ILAT (if<br>enabled) |
|----------|---------|---------|----------------------|----------|---------------------------------|
| Anything | NA      | NA      | NA                   | b#01     | Set                             |

## WATCHDOG Events

**Table 15-18:** Startup Event

| TMR_PER               | TMR_DLY               | TMR_WID                       | TMR_WID<br>+ TMR_DLY | ERR_TYPE  | TIMER_STAT_ILAT (if<br>enabled) |
|-----------------------|-----------------------|-------------------------------|----------------------|-----------|---------------------------------|
| =< Allowed<br>MIN[12] | Anything <<br>PERIOD  | NA                            | NA                   | b#01      | Set                             |
| > Allowed MIN         | Anything <<br>PERIOD  | NA                            | NA                   | Unchanged | Unchanged                       |
| > Allowed MIN         | Anything >=<br>PERIOD | Refer to WATCHDOG Mode tables |                      |           |                                 |

**Table 15-19:** Rollover Event

| TMR_PER               | TMR_DLY               | TMR_WID                       | TMR_WID +<br>TMR_DLY | ERR_TYPE  | TIMER_STAT_ILAT (if<br>enabled) |
|-----------------------|-----------------------|-------------------------------|----------------------|-----------|---------------------------------|
| =< Allowed<br>MIN[10] | Anything <<br>PERIOD  | NA                            | NA                   | b#01      | Set                             |
| > Allowed MIN         | Anything              | NA                            | NA                   | Unchanged | Unchanged                       |
| > Allowed MIN         | Anything >=<br>PERIOD | Refer to WATCHDOG Mode tables |                      |           |                                 |

**Table 15-20:** Overflow Event

| TMR_PER  | TMR_DLY  | TMR_WID | TMR_WID +<br>TMR_DLY | ERR_TYPE | TIMER_STAT_ILAT (if<br>enabled) |
|----------|----------|---------|----------------------|----------|---------------------------------|
| Anything | Anything | NA      | NA                   | b#01     | Set                             |

## TIMER Register Descriptions

General Purpose Timer Block (TIMER) contains the following registers.

**Table 15-21:** TIMER Module Registers

| Name               | Description                       |
|--------------------|-----------------------------------|
| TIMER_REVID        | Revision ID Register              |
| TIMER_RUN          | Run Register                      |
| TIMER_RUN_SET      | Run Set Register                  |
| TIMER_RUN_CLR      | Run Clear Register                |
| TIMER_STOP_CFG     | Stop Configuration Register       |
| TIMER_STOP_CFG_SET | Stop Configuration Set Register   |
| TIMER_STOP_CFG_CLR | Stop Configuration Clear Register |
| TIMER_DATA_IMSK    | Data Interrupt Mask Register      |
| TIMER_STAT_IMSK    | Status Interrupt Mask Register    |
| TIMER_TRG_MSK      | Trigger Master Mask Register      |
| TIMER_TRG_IE       | Trigger Slave Enable Register     |
| TIMER_DATA_ILAT    | Data Interrupt Latch Register     |
| TIMER_STAT_ILAT    | Status Interrupt Latch Register   |
| TIMER_ERR_TYPE     | Error Type Status Register        |
| TIMER_BCAST_PER    | Broadcast Period Register         |
| TIMER_BCAST_WID    | Broadcast Width Register          |
| TIMER_BCAST_DLY    | Broadcast Delay Register          |
| TIMER_TMRn_CFG     | Timer n Configuration Register    |
| TIMER_TMRn_CNT     | Timer n Counter Register          |
| TIMER_TMRn_PER     | Timer n Period Register           |

**Table 15-21:** TIMER Module Registers (Continued)

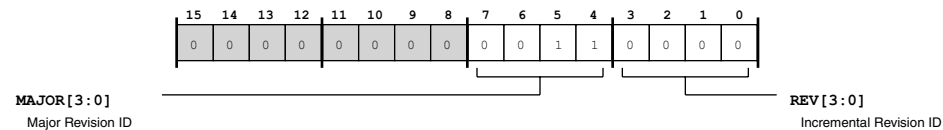
| Name           | Description            |
|----------------|------------------------|
| TIMER_TMRn_WID | Timer n Width Register |
| TIMER_TMRn_DLY | Timer n Delay Register |

## Revision ID Register

The `TIMER_REVID` holds the TIMER module revision information.

**TIMER\_REVID: Revision ID Register - R/NW**

Reset = 0x0030

**Figure 15-10:** TIMER\_REVID Register Diagram**Table 15-22:** TIMER\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 7:4     | MAJOR    | Major Revision ID.       |
| 3:0     | REV      | Incremental Revision ID. |

## Run Register

The `TIMER_RUN` allows all timers to be enabled simultaneously, permitting them to run synchronously. For each timer, there is a single start/stop control bit. Writing a 1 to this bit starts the corresponding timer; writing a 0 stops the timer with mechanism specified in the timer stop configuration `TIMER_STOP_CFG` register.

The start/stop control bits can be set/reset individually or in any combination. While starting or stopping one particular timer directly with this register, software must perform a read-modify write, so the bits corresponding to other timers remain unchanged. To avoid this need, software can use the `TIMER_RUN_CLR` register.

Reading the `TIMER_RUN` register shows the start status for the corresponding timer. A 1 indicates that the timer is running.

If a timer is in run state (corresponding run bit is =1), a software write of 1 in this bit does not have any effect on the timer state. The write does not result in restarting the timer.

Note that the `TIMER_RUN` register is not used in PININT mode. PININT mode starts as soon as the `TIMER_TMRn_CFG.TMODE` bits are set to 111.

**TIMER\_RUN: Run Register - R/W**

Reset = 0x0000

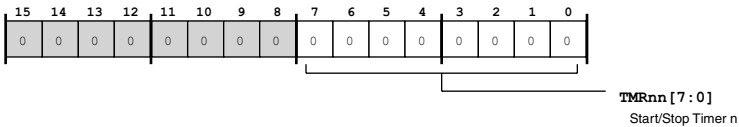


**Figure 15-11:** TIMER\_RUN Register Diagram

**Table 15-23:** TIMER\_RUN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Start/Stop Timer n.<br>For all <code>TIMER_RUN.TMRnn</code> bits, write =0 for stop, and write =1 for start. Read =1 when timer is running. |

Run Set Register

The `TIMER_RUN_SET` register is an alias register, providing a mechanism to set a specific start/stop bit in the `TIMER_RUN` register without affecting other bits in `TIMER_RUN`. To start a particular timer, software must write a 1 into the corresponding `TIMER_RUN_SET` bit. Writing a zero has no effect. For an example, to start timer 3 without affecting any other timer, write 0x0008 into `TIMER_RUN_SET`. Because `TIMER_RUN_SET` is a write-only register, the result of any write to this register must be checked by reading the `TIMER_RUN` register. A read of the `TIMER_RUN_SET` returns 0x0000.

**TIMER\_RUN\_SET: Run Set Register - R/WA**

Reset = 0x0000

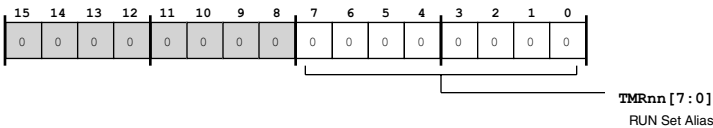


**Figure 15-12:** TIMER\_RUN\_SET Register Diagram

**Table 15-24:** TIMER\_RUN\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | RUN Set Alias.<br>For all <code>TIMER_RUN_SET.TMRnn</code> bits, write =0 has no effect, and write =1 for start (setting the corresponding start/stop bit in the <code>TIMER_RUN</code> register). Using <code>TIMER_RUN_SET</code> to set start/stop bits permits starting specific timers without influencing the run status of other timers. |

## Run Clear Register

The `TIMER_RUN_CLR` register is an alias register, providing a mechanism to clear a specific start/stop bit in the `TIMER_RUN` register without affecting other bits in `TIMER_RUN`. To stop a particular timer, software must write a 1 into the corresponding `TIMER_RUN_CLR` bit. Writing a 0 has no effect. Because `TIMER_RUN_CLR` is a write-only register, the result of any write to this register must be checked by reading the `TIMER_RUN` register. A read of the `TIMER_RUN_CLR` returns 0x0000.

Note that the stopping mechanism of a timer may be abrupt or graceful (after completion of current waveform period) depending on the selection in the `TIMER_STOP_CFG` register.

**TIMER\_RUN\_CLR: Run Clear Register - R/WA**

Reset = 0x0000

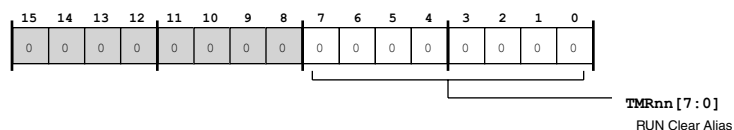


**Figure 15-13:** TIMER\_RUN\_CLR Register Diagram

**Table 15-25:** TIMER\_RUN\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | RUN Clear Alias.<br>For all <code>TIMER_RUN_CLR.TMRnn</code> bits, write =0 has no effect, and write =1 for stop (clearing the corresponding in start/stop bit in the <code>TIMER_RUN</code> register). Using <code>TIMER_RUN_CLR</code> to clear start/stop bits permits stopping specific timers without influencing run status of other timers. |

## Stop Configuration Register

The `TIMER_STOP_CFG` selects the stop mode for each timer. Timers may be stopped abruptly (immediate halt - all modes) or gracefully in PWMOUT modes (single pulse and continuous). The halt is achieved through either a write =0 to the corresponding bit in `TIMER_RUN` or a write =1 to the corresponding bit in `TIMER_RUN_CLR`. A read of `TIMER_STOP_CFG` returns the last value written.

**TIMER\_STOP\_CFG: Stop Configuration Register - R/W**

Reset = 0x0000

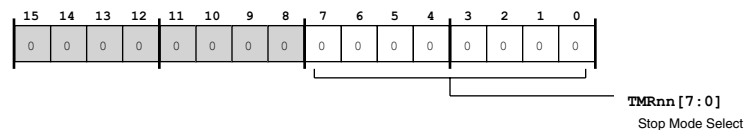


**Figure 15-14:** TIMER\_STOP\_CFG Register Diagram

**Table 15-26:** TIMER\_STOP\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Stop Mode Select.<br>For all <code>TIMER_STOP_CFG.TMRnn</code> bits, write =0 for graceful termination (PWMOUT modes only), and write =1 for abrupt (immediate halt) on stop. |

## Stop Configuration Set Register

This is an alias register, providing a mechanism to set a specific bit in the `TIMER_STOP_CFG` register without affecting other bits in `TIMER_STOP_CFG`. To set a bit in `TIMER_STOP_CFG`, software must write a 1 to the corresponding bit of `TIMER_STOP_CFG_SET`. Writing a zero has no effect. Because `TIMER_STOP_CFG_SET` is a write-only register, the result of any write to this register must be checked by reading the `TIMER_STOP_CFG` register. A read of the `TIMER_STOP_CFG_SET` returns 0x0000.

**TIMER\_STOP\_CFG\_SET: Stop Configuration Set Register - R/WA**

Reset = 0x0000

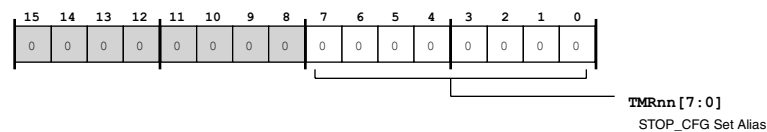


**Figure 15-15:** TIMER\_STOP\_CFG\_SET Register Diagram



**Table 15-27:** TIMER\_STOP\_CFG\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | STOP_CFG Set Alias.<br>For all TIMER_STOP_CFG_SET.TMRnn bits, write =0 has no effect, and write =1 for abrupt stop (setting the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_SET to set stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers. |

## Stop Configuration Clear Register

This is an alias register, providing a mechanism to clear a specific bit in the TIMER\_STOP\_CFG register without affecting other bits in TIMER\_STOP\_CFG. To clear a bit in TIMER\_STOP\_CFG, software must write a 1 to the corresponding bit of TIMER\_STOP\_CFG\_CLR. Writing a zero has no effect. Because TIMER\_STOP\_CFG\_CLR is a write-only register, the result of any write to this register must be checked by reading the TIMER\_STOP\_CFG register. A read of the TIMER\_STOP\_CFG\_CLR returns 0x0000.

**TIMER\_STOP\_CFG\_CLR: Stop Configuration Clear Register - R/WA**

Reset = 0x0000

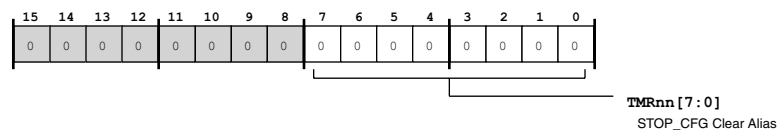


**Figure 15-16:** TIMER\_STOP\_CFG\_CLR Register Diagram

**Table 15-28:** TIMER\_STOP\_CFG\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | STOP_CFG Clear Alias.<br>For all TIMER_STOP_CFG_CLR.TMRnn bits, write =0 has no effect, and write =1 for graceful stop in PWMOUT modes (clearing the corresponding stop mode select bit in the TIMER_STOP_CFG register). Using TIMER_STOP_CFG_CLR to clear stop mode bits permits configuring specific timers without influencing the stop mode configuration of other timers. |

## Data Interrupt Mask Register

Each timer may generate a unique processor data interrupt request signal. The `TIMER_DATA_IMSK` contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the `TIMER_DATA_IMSK` register is `0xFFFF`, masking these interrupts after reset.

**TIMER\_DATA\_IMSK: Data Interrupt Mask Register - R/W**

Reset = 0x00ff

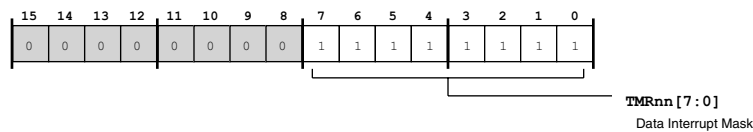


**Figure 15-17:** TIMER\_DATA\_IMSK Register Diagram

**Table 15-29:** TIMER\_DATA\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Data Interrupt Mask.<br>For all <code>TIMER_DATA_IMSK.TMRnn</code> bits, write =0 unmask (enables) the corresponding data interrupt request, and write =1 masks (disables) the corresponding data interrupt request. |

## Status Interrupt Mask Register

While each timer may generate a status interrupt request, these requests are OR'ed to generate a single status interrupt signal to the System Event Controller. The `TIMER_STAT_IMSK` contains an interrupt mask for these requests, masking (disabling) or unmasking (enabling) the interrupts as programmed. The reset value of the `TIMER_STAT_IMSK` register is `0xFFFF`, masking these interrupts after reset.

**TIMER\_STAT\_IMSK: Status Interrupt Mask Register - R/W**

Reset = 0x00ff

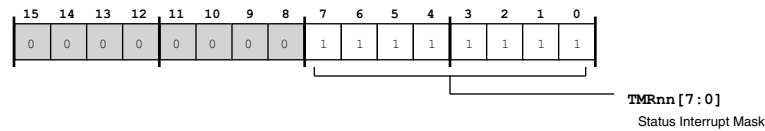


**Figure 15-18:** TIMER\_STAT\_IMSK Register Diagram

**Table 15-30:** TIMER\_STAT\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Status Interrupt Mask.<br>For all <code>TIMER_STAT_IMSK.TMRnn</code> bits, write =0 unmask (enables) the corresponding status interrupt request, and write =1 masks (disables) the corresponding status interrupt request. |

## Trigger Master Mask Register

As a trigger master, each timer can generate a unique data trigger pulse signal. The `TIMER_TRG_MSK` contains a trigger mask for these outputs, masking (disabling) or unmasking (enabling) the triggers as programmed. The reset value of the `TIMER_TRG_MSK` register is `0xFFFF`, masking these triggers after reset.

**TIMER\_TRG\_MSK: Trigger Master Mask Register - R/W**

Reset = `0x00ff`

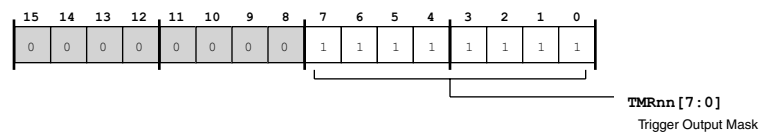


**Figure 15-19:** TIMER\_TRG\_MSK Register Diagram

**Table 15-31:** TIMER\_TRG\_MSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Trigger Output Mask.<br>For all <code>TIMER_TRG_MSK.TMRnn</code> bits, write =0 unmask (enables) the corresponding data trigger output, and write =1 masks (disables) the corresponding data trigger output. |

## Trigger Slave Enable Register

As a trigger slave, each timer can generate a unique data trigger pulse signal. The `TIMER_TRG_IE` contains trigger input enable bits for these signals, disabling or enabling the triggers as programmed. The reset value of the `TIMER_TRG_IE` register is `0xFFFF`, masking these triggers after reset.

TIMER\_TRG\_IE: Trigger Slave Enable Register - R/W

Reset = 0x0000

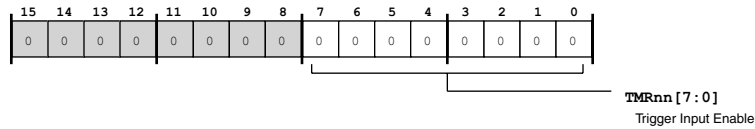


Figure 15-20: TIMER\_TRG\_IE Register Diagram

Table 15-32: TIMER\_TRG\_IE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Trigger Input Enable.<br>For all TIMER_TRG_IE.TMRnn bits, write =0 disables the corresponding trigger input, and write =1 enables the corresponding trigger input. |

Data Interrupt Latch Register

The TIMER\_DATA\_ILAT holds the latched interrupt status for interrupt requests that have been unmasked (enabled) by the TIMER\_DATA\_IMSK register and generated according to the conditions selected by the TIMER\_TMRn\_CFG.IRQMODE bits. If a bit in TIMER\_DATA\_ILAT is already set and the corresponding interrupt is masked in TIMER\_DATA\_IMSK, the latch holds its old value, leaving the interrupt asserted until it is reset by software with a W1C operation.

Note that interrupt service routines (ISRs) should clear the appropriate bits in TIMER\_DATA\_ILAT before returning from the ISR, to ensure that the interrupt is not re-issued. To make sure that no timer event is missed, the latch should be reset at the very beginning of the ISR when in EXTCLK mode.

TIMER\_DATA\_ILAT: Data Interrupt Latch Register - R/W

Reset = 0x0000

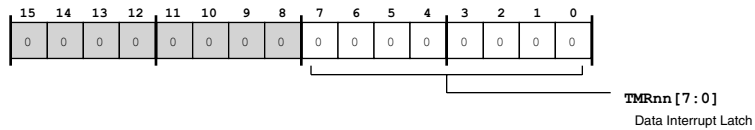


Figure 15-21: TIMER\_DATA\_ILAT Register Diagram

**Table 15-33:** TIMER\_DATA\_ILAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Data Interrupt Latch.<br>For all <code>TIMER_DATA_ILAT.TMRnn</code> bits, status of =0 indicates no interrupt is latched, and status of =1 indicates a latched interrupt (indicating an unmasked interrupt request from a timer with a condition matching the one selected with corresponding <code>TIMER_TMRn_CFG.IRQMODE</code> bit has occurred). |

## Status Interrupt Latch Register

The `TIMER_STAT_ILAT` holds the latched interrupt status for error interrupts, indicating a timer overflow condition or indicating that prohibited programming has occurred for a timer. These interrupt status bits are sticky and are W1C. The bits in the `TIMER_STAT_ILAT` register provide information regarding each timer interrupt source.

**TIMER\_STAT\_ILAT: Status Interrupt Latch Register - R/W**

Reset = 0x0000

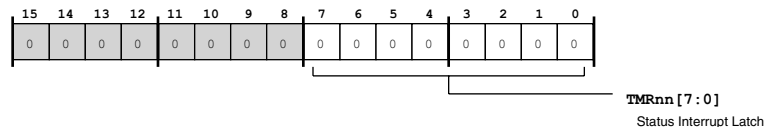


**Figure 15-22:** TIMER\_STAT\_ILAT Register Diagram

**Table 15-34:** TIMER\_STAT\_ILAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | TMRnn    | Status Interrupt Latch.<br>For all <code>TIMER_STAT_ILAT.TMRnn</code> bits, status of 0 indicates no error interrupt is latched, and status of 1 indicates a timer counter overflow or programming error interrupt is latched. |

## Error Type Status Register

The `TIMER_ERR_TYPE` contains Error Type status bits for each timer. These bits indicate the type of error (if any) in a running timer. This register is read-only. These status bits are cleared at reset and when a particular timer is enabled.

Each time an error interrupt is latched in `TIMER_STAT_ILAT`, the corresponding `TERRx` bits in `TIMER_ERR_TYPE` are loaded with a code that identifies the type of error that was detected. This status value is held until the next error or until a particular timer is restarted. No bus error is generated if a write is performed on `TIMER_ERR_TYPE`.

**TIMER\_ERR\_TYPE: Error Type Status Register - R/NW**

Reset = 0x0000 0000

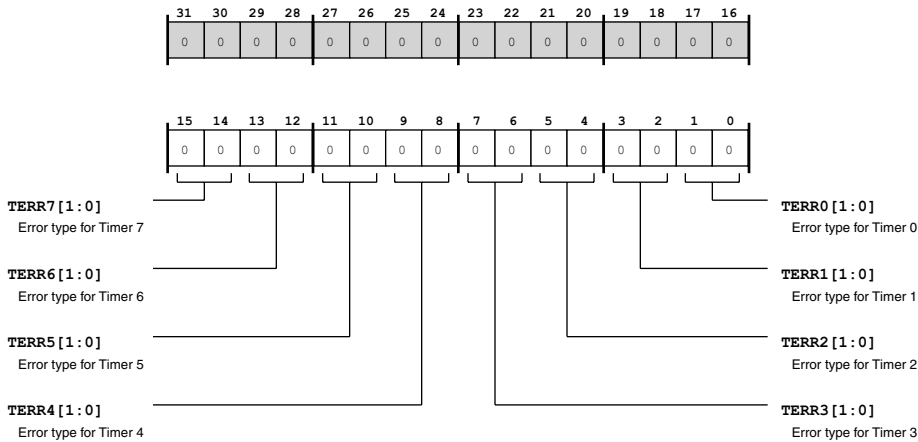


**Figure 15-23:** `TIMER_ERR_TYPE` Register Diagram

**Table 15-35:** `TIMER_ERR_TYPE` Register Fields

| Bit No. | Bit Name | Description/Enumeration |                                       |
|---------|----------|-------------------------|---------------------------------------|
| 15:14   | TERR7    | Error type for Timer 7. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |
| 13:12   | TERR6    | Error type for Timer 6. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |

**Table 15-35:** TIMER\_ERR\_TYPE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                       |
|---------|----------|-------------------------|---------------------------------------|
| 11:10   | TERR5    | Error type for Timer 5. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |
| 9:8     | TERR4    | Error type for Timer 4. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |
| 7:6     | TERR3    | Error type for Timer 3. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |
| 5:4     | TERR2    | Error type for Timer 2. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |

Table 15-35: TIMER\_ERR\_TYPE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                       |
|---------|----------|-------------------------|---------------------------------------|
| 3:2     | TERR1    | Error type for Timer 1. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |
| 1:0     | TERR0    | Error type for Timer 0. |                                       |
|         |          | 0                       | No Error                              |
|         |          | 1                       | Counter Overflow Error                |
|         |          | 2                       | PER Register Programming Error        |
|         |          | 3                       | WID or DLY Register Programming Error |

Broadcast Period Register

For timers with `TIMER_TMRn_CFG.BPEREN` enabled, a write to `TIMER_BCAST_PER` concurrently updates the period (`TIMER_TMRn_PER`) registers of only those timers. A read of `TIMER_BCAST_PER` returns `0x00000000`, and no bus error is generated. To read back a written value, read that TMR's `TIMER_TMRn_PER` register.

TIMER\_BCAST\_PER: Broadcast Period Register - R/WA

Reset = 0x0000 0000

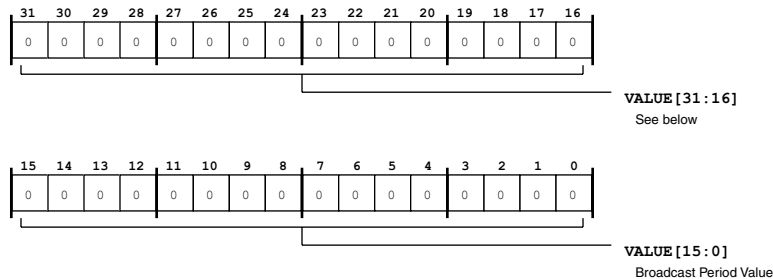


Figure 15-24: TIMER\_BCAST\_PER Register Diagram



**Table 15-36:** TIMER\_BCAST\_PER Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Broadcast Period Value. |

## Broadcast Width Register

For timers with `TIMER_TMRn_CFG.BWIDEN` enabled, a write to `TIMER_BCAST_WID` concurrently updates the width (`TIMER_TMRn_WID`) registers of only those timers. A read of `TIMER_BCAST_WID` returns `0x00000000`, and no bus error is generated. To read back a written value, read that TMR's `TIMER_TMRn_WID` register.

### TIMER\_BCAST\_WID: Broadcast Width Register - R/WA

Reset = 0x0000 0000

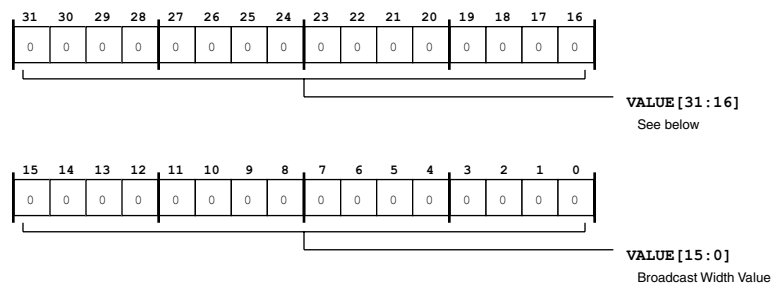


**Figure 15-25:** TIMER\_BCAST\_WID Register Diagram

**Table 15-37:** TIMER\_BCAST\_WID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Broadcast Width Value.  |

## Broadcast Delay Register

For timers with `TIMER_TMRn_CFG.BDLYEN` enabled, a write to `TIMER_BCAST_DLY` concurrently updates the delay (`TIMER_TMRn_DLY`) registers of only those timers. A read of `TIMER_BCAST_DLY` returns `0x00000000`, and no bus error is generated. To read back a written value, read that TMR's `TIMER_TMRn_DLY` register.

TIMER\_BCAST\_DLY: Broadcast Delay Register - R/WA

Reset = 0x0000 0000

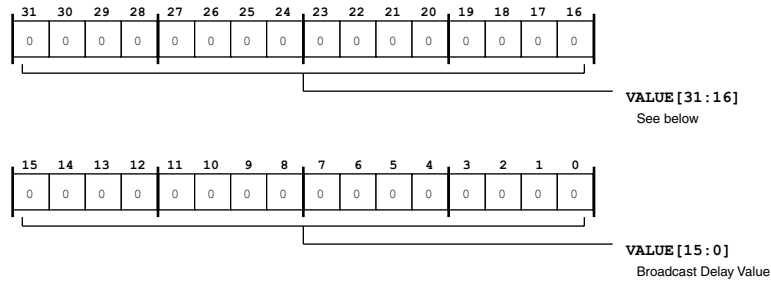


Figure 15-26: TIMER\_BCAST\_DLY Register Diagram

Table 15-38: TIMER\_BCAST\_DLY Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Broadcast Delay Value.  |

Timer n Configuration Register

Each timer has a `TIMER_TMRn_CFG` register that specifies its operating mode. Only write to a `TIMER_TMRn_CFG` register when the corresponding timer is not running.

After disabling a timer operating in `PWMOUT` mode, verify that the timer has stopped running by checking the start/stop status of the timer in the `TIMER_RUN` register before writing to the timer's `TIMER_TMRn_CFG` register.

Note that a timer's `TIMER_TMRn_CFG` register may be read at any time.

**TIMER\_TMRn\_CFG: Timer n Configuration Register - R/W**

Reset = 0x0000

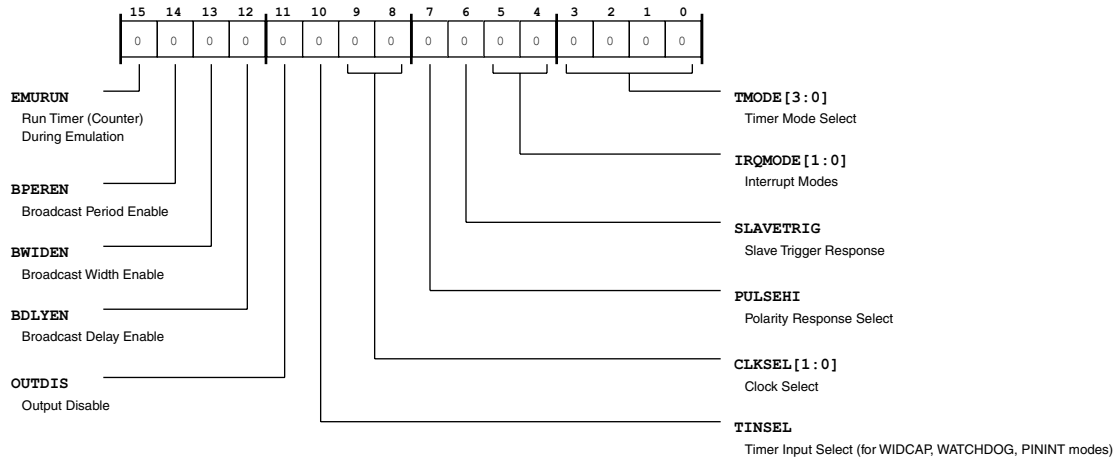


**Figure 15-27:** TIMER\_TMRn\_CFG Register Diagram

**Table 15-39:** TIMER\_TMRn\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 15      | EMURUN   | Run Timer (Counter) During Emulation. |
|         |          | 0 Stop Timer During Emulation         |
|         |          | 1 Run Timer During Emulation          |
| 14      | BPEREN   | Broadcast Period Enable.              |
|         |          | 0 Disable Broadcast to PER Register   |
|         |          | 1 Enable Broadcast to PER Register    |
| 13      | BWIDEN   | Broadcast Width Enable.               |
|         |          | 0 Disable Broadcast to WID Register   |
|         |          | 1 Enable Broadcast to WID Register    |
| 12      | BDLYEN   | Broadcast Delay Enable.               |
|         |          | 0 Disable Broadcast to DLY Register   |
|         |          | 1 Enable Broadcast to DLY Register    |

**Table 15-39:** TIMER\_TMRn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                  |                                                                                                                                                                                                                                           |
|---------|----------|----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | OUTDIS   | Output Disable.                                          |                                                                                                                                                                                                                                           |
|         |          | 0                                                        | Enable TMR pin output buffer                                                                                                                                                                                                              |
|         |          | 1                                                        | Disable TMR pin output buffer                                                                                                                                                                                                             |
| 10      | TINSEL   | Timer Input Select (for WIDCAP, WATCHDOG, PININT modes). |                                                                                                                                                                                                                                           |
|         |          | 0                                                        | Use TMR pin input                                                                                                                                                                                                                         |
|         |          | 1                                                        | Use TMR Alternate Capture Input                                                                                                                                                                                                           |
| 9:8     | CLKSEL   | Clock Select.                                            |                                                                                                                                                                                                                                           |
|         |          | 0                                                        | Use SCLK                                                                                                                                                                                                                                  |
|         |          | 1                                                        | Use TMR_ALT_CLK0 as the TMR clock                                                                                                                                                                                                         |
|         |          | 3                                                        | Use TMR_ALT_CLK1 as the TMR clock                                                                                                                                                                                                         |
| 7       | PULSEHI  | Polarity Response Select.                                |                                                                                                                                                                                                                                           |
|         |          | 0                                                        | Negative Response/Pulse<br>Negative Edge Response on TMR pin or Alternate Capture pin causes interrupt (PININT, EXTCLK modes) Negative Action Pulse on TMR pin or Alternate Capture pin causes interrupt (PWMOUT, WATCHDOG, WIDCAP modes) |
|         |          | 1                                                        | Positive Response/Pulse<br>Positive Edge Response on TMR pin or Alternate Capture pin causes interrupt (PININT, EXTCLK modes) Positive Action Pulse on TMR pin or Alternate Capture pin causes interrupt (PWMOUT, WATCHDOG, WIDCAP modes) |

**Table 15-39:** TIMER\_TMRn\_CFG Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                           |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | SLAVETRIG | Slave Trigger Response.<br>Note that the trigger pulse has no effect (to stop or start the timer) if the timer is already in the requested state. |
|         |           | 0 Pulse stops timer if it is running                                                                                                              |
|         |           | 1 Pulse starts timer if it is stopped                                                                                                             |

**Table 15-39:** TIMER\_TMRn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | IRQMODE  | <p>Interrupt Modes.</p> <p>The <code>TIMER_TMRn_CFG.IRQMODE</code> selects the interrupt request mode. Note that any mismatched combination of <code>TIMER_TMRn_CFG.IRQMODE</code> and <code>TIMER_TMRn_CFG.TMODE</code> results in no interrupt being generated.</p> <p>Also note that in WIDCAP modes, the position of the interrupt is controlled with the <code>TIMER_TMRn_CFG.TMODE</code> bit, and the <code>TIMER_TMRn_CFG.IRQMODE</code> bit is ignored.</p> |
|         |          | <p>0</p> <p>Active Edge Mode</p> <p>The timer generates an interrupt at every active edge. The active edge polarity depends on the state of the <code>TIMER_TMRn_CFG.PULSEHI</code> bit). Valid for PININT mode only.</p>                                                                                                                                                                                                                                            |
|         |          | <p>1</p> <p>Delay Expired Mode</p> <p>The timer generates an interrupt when the value reaches the value in the register. This mode is valid for all PWMOUT modes.</p>                                                                                                                                                                                                                                                                                                |
|         |          | <p>2</p> <p>Width Plus Delay Expired Mode</p> <p>The timer generates an interrupt when the value reaches the value in the register plus the value in the register. (PWMOUT modes only).</p> <p>The timer generates an interrupt if the de-asserting edge is within the specified window. (WATCHDOG modes only.)</p>                                                                                                                                                  |
|         |          | <p>3</p> <p>Period Expired Mode</p> <p>The timer generates an interrupt when the value reaches the value in the register. (Continuous PWMOUT and EXTCLK modes only.)</p> <p>The timer generates an interrupt if the de-asserting edge is outside the specified window.(WATCHDOG modes only.)</p>                                                                                                                                                                     |

**Table 15-39:** TIMER\_TMRn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------|
| 3:0     | TMODE    | Timer Mode Select.                                                                      |
|         |          | 0000 - 0111 Idle Mode                                                                   |
|         |          | 8 Period Watchdog Mode                                                                  |
|         |          | 9 Width Watchdog Mode                                                                   |
|         |          | 10 Width Capture Asserted Mode<br>Measurement report at asserting edge of waveform      |
|         |          | 11 Width Capture Deasserted Mode<br>Measurement report at de-asserting edge of waveform |
|         |          | 12 Continuous PWMOUT mode                                                               |
|         |          | 13 Single pulse PWMOUT mode                                                             |
|         |          | 14 EXTCLK mode                                                                          |
|         |          | 15 PININT (pin interrupt) mode                                                          |

## Timer n Counter Register

The `TIMER_TMRn_CNT` holds the current timer count. After enabling, the count is re-initialized to either 0x0 or 0x1, depending on the configuration and mode. The `TIMER_TMRn_CNT` is read only and may be read at any time (whether the timer is running or stopped). Reading `TIMER_TMRn_CNT` returns an atomic 32-bit value.

Depending on the timer operation mode, the counter increment can be clocked by a number of sources, including SCLK, the TMR or Alternate Capture input pins, `TMR_ALT_CLK0`, or `TMR_ALT_CLK1`. The counter retains its value after the timer is disabled.

TIMER\_TMRn\_CNT: Timer n Counter Register - R/NW

Reset = 0x0000 0001

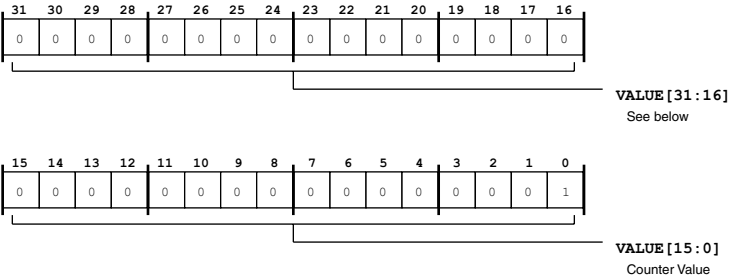


Figure 15-28: TIMER\_TMRn\_CNT Register Diagram

Table 15-40: TIMER\_TMRn\_CNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Counter Value.          |

Timer n Period Register

The TIMER\_TMRn\_PER holds the period value for the corresponding timer. Has different uses depending on the timer mode.

TIMER\_TMRn\_PER: Timer n Period Register - R/W

Reset = 0x0000 0000

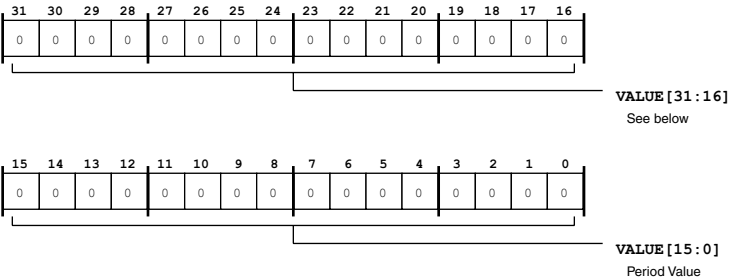


Figure 15-29: TIMER\_TMRn\_PER Register Diagram

Table 15-41: TIMER\_TMRn\_PER Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Period Value.           |



Timer n Width Register

The `TIMER_TMRn_WID` holds the width value for the corresponding timer. Has different uses depending on the timer mode.

TIMER\_TMRn\_WID: Timer n Width Register - R/W

Reset = 0x0000 0000

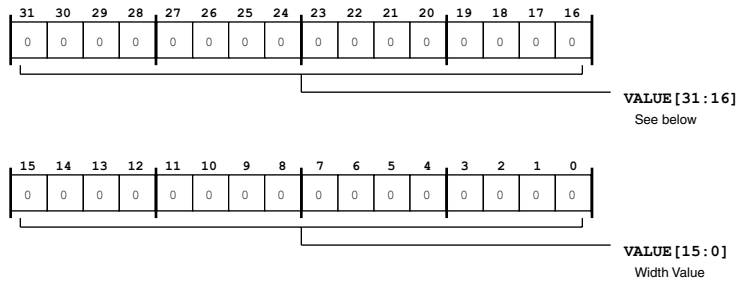


Figure 15-30: TIMER\_TMRn\_WID Register Diagram

Table 15-42: TIMER\_TMRn\_WID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Width Value.            |

Timer n Delay Register

The `TIMER_TMRn_DLY` holds the delay value for the corresponding timer. Has different uses depending on the timer mode.

TIMER\_TMRn\_DLY: Timer n Delay Register - R/W

Reset = 0x0000 0000

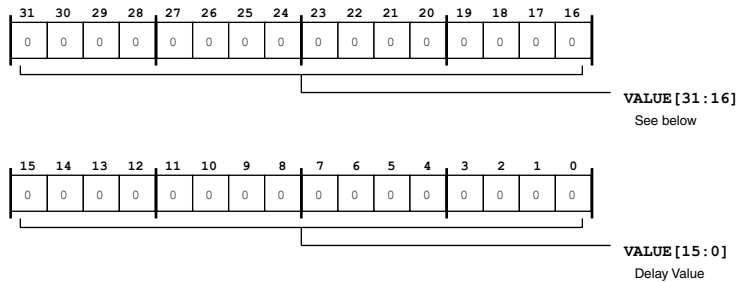


Figure 15-31: TIMER\_TMRn\_DLY Register Diagram

**Table 15-43:** TIMER\_TMRn\_DLY Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Delay Value.            |

# 16 Watchdog Timer (WDOG)

The processor includes a 32-bit timer for each core that can be used to implement a software watchdog function. A software watchdog can improve system reliability by generating an event to the processor core if the watchdog expires before being updated by software. The watchdog timers are clocked by the system clock (SCLK).

## WDOG Features

The watchdog timer has the following features.

- Two identical 32-bit watchdog timers
- 8-bit disable bit pattern
- Generates a general-purpose event for the core.

Typically, the watchdog timer is used to supervise stability of the system software. When used in this way, software reloads the watchdog timer in a regular manner so that the downward counting timer never expires (never becomes 0). An expiring timer then indicates that system software might be out of control. At this point based on the GP event generated by WDT, it is often better to reset and reboot the system using the Reset Control Unit.

For easier debugging, the watchdog timer does not decrement (even if enabled) when the processor is in emulation mode.

**NOTE:** The emulation event is controlled by the SDU (System Debug Unit). Please refer to the SDU chapter for more details on this.

## Watchdog Timer Functional Description

When enabled, the 32-bit watchdog timer counts downward every SCLK cycle. When the count becomes 0, the expiry event is generated. This generates an GP event to the processor. When the event is generated, the core and the peripherals need to be reset using the software. The counter value can be read through the 32-bit WDOG\_STAT register. The WDOG\_STAT register cannot, however, be written directly. Rather, software writes the watchdog period value into the 32-bit WDOG\_CNT register before the watchdog is enabled. Once the watchdog is started, the period value cannot be altered.

WDOG Register List

Table 16-1: WDOG Module Registers

| Name      | Description                    |
|-----------|--------------------------------|
| WDOG_CTL  | Control Register               |
| WDOG_CNT  | Count Register                 |
| WDOG_STAT | Watchdog Timer Status Register |

ADSP-BF60x WDOG Interrupt List

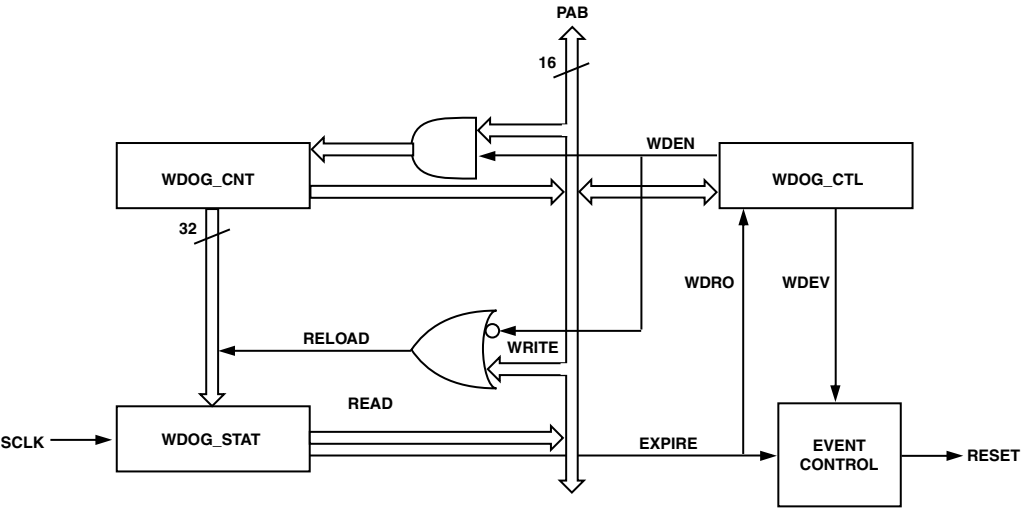
Table 16-2: ADSP-BF60x WDOG Interrupt List

| Description      | ID | DMA Channel | Sensitivity |
|------------------|----|-------------|-------------|
| WDOG0 Expiration | 2  |             | LEVEL       |
| WDOG1 Expiration | 3  |             | LEVEL       |

WDOG Block Diagram

The figure below shows the detailed watchdog timer block diagram.

Watchdog Timer Block Diagram



## Internal Interface

The watchdog timer does not directly interact with any pins of the chip.

## External Interface

The watchdog timer is clocked by the system clock (SCLK) and its registers are accessed through the 16-bit peripheral MMR access bus. The 32-bit WDOG\_CNT and WDOG\_STAT registers must always be accessed by 32-bit read/write operations. Hardware ensures that those accesses are atomic. When the counter expires, the GP expiry event is generated.

## WDOG Configuration

### *PREREQUISITE:*

To start the watchdog timer, use the following procedure

1. Set the count value for the watchdog timer by writing the count value into the Watchdog Count register (WDOG\_CNT). Note that loading the WDOG\_CNT register while the timer is not enabled also pre-loads the WDOG\_STAT register.
2. Enable the watchdog timer by writing to the WDOG\_CTL.WDEN bit field. The watchdog timer then begins counting down, decrementing the value in the WDOG\_STAT register. When the WDOG\_STAT reaches 0, the expiry event is generated.

*ADDITIONAL INFORMATION:* To prevent the event from being generated, software must reload the count value from the WDOG\_CNT register to the WDOG\_STAT register by executing a write (of any value) to WDOG\_STAT, or must disable the watchdog timer in the WDOG\_CTL register before the watchdog timer expires.

- a. If software does not serve the watchdog in time, the WDOG\_STAT register continues decrementing until it reaches 0. When watchdog decrements and reaches 0, it generates a GP interrupt if enabled and the software can perform core and/or system reset.

*ADDITIONAL INFORMATION:* Once the counter stops reaches 0, it stops decrementing and remains at 0. Additionally, the WDRO latch bit in the WDOG\_CTL register is set.

- b. If the watchdog is enabled with a zero value loaded to the counter and the WDRO bit was cleared, the WDRO bit of the watchdog control register is set immediately and the counter remains at zero without further decrements. If, however, the WDRO bit was set by the time the watchdog is enabled, the counter decrements to 0xFFFF FFFF and continues operation.

*ADDITIONAL INFORMATION:* Software can disable the watchdog timer only by writing a 0xAD value (WDDIS) to the WDEN field in the WDOG\_CTL register.

## WDOG Register Descriptions

Watch Dog Timer Unit (WDOG) contains the following registers.

**Table 16-3:** WDOG Module Registers

| Name      | Description                    |
|-----------|--------------------------------|
| WDOG_CTL  | Control Register               |
| WDOG_CNT  | Count Register                 |
| WDOG_STAT | Watchdog Timer Status Register |

### Control Register

The WDOG\_CTL register controls the watch dog timer. This register supports enabling/disabling the watch dog timer and supports checking the timer rollover status. Note that when the processor is in emulation mode, the watch dog timer counter will not decrement even if it is enabled.

**WDOG\_CTL: Control Register - R/W**

Reset = 0x0000 0ad0

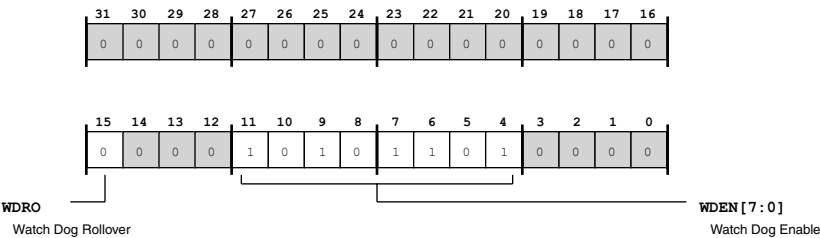


**Figure 16-1:** WDOG\_CTL Register Diagram

**Table 16-4:** WDOG\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | WDRO     | Watch Dog Rollover.<br>Software can determine whether the timer has rolled over by interrogating the WDOG_CTL.WDRO status bit. This is a sticky bit that is set whenever the watch dog timer count reaches 0 and cleared only by disabling the watch dog timer and then writing a 1 to the bit. |
|         |          | 0 WDT has not expired                                                                                                                                                                                                                                                                           |
|         |          | 1 WDT has expired                                                                                                                                                                                                                                                                               |
| 11:4    | WDEN     | Watch Dog Enable.<br>The WDOG_CTL.WDEN field is used to enable and disable the watch dog timer. Writing any value other than the disable value into this field enables the watch dog timer. This multi-bit disable key minimizes the chance of inadvertently disabling the watch dog timer.     |
|         |          | 173 Counter Disabled<br>All other values - counter enabled                                                                                                                                                                                                                                      |

## Count Register

The WDOG\_CNT register holds the programmable, unsigned count value. A valid write to this register also pre-loads the WDOG counter. For added safety, the WDOG\_CNT register can be updated only when the WDOG timer is disabled. A write to the WDOG\_CNT register while the timer is enabled does not modify the contents of this register. This register must be accessed with 32-bit read/writes only.

### WDOG\_CNT: Count Register - R/W

Reset = 0x0000 0000

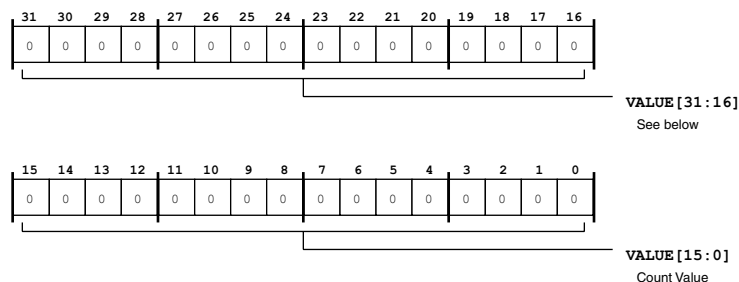


**Figure 16-2:** WDOG\_CNT Register Diagram

**Table 16-5:** WDOG\_CNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Count Value.            |

## Watchdog Timer Status Register

The WDOG\_STAT contains the current count value of the watch dog timer. Reads of this register return the current count value. When the watch dog timer is enabled, WDOG\_STAT is decremented by 1 on each SCLK cycle. When count value reaches 0, the watch dog timer stops counting, and the expiry event is generated. WDOG\_STAT is a 32-bit unsigned system MMR that must be accessed with 32-bit reads and writes.

Values cannot be stored directly in WDOG\_STAT, but are instead copied from the WDOG\_CNT register. This copy process can happen in two ways:

- While the watch dog timer is disabled, writing the WDOG\_CNT register pre-loads the WDOG\_STAT register.
- While the watch dog timer is enabled, writing the WDOG\_STAT register loads it with the value in WDOG\_CNT.

When the processor executes a write (of an arbitrary value) to WDOG\_STAT, the value in WDOG\_CNT is copied into WDOG\_STAT. Typically, software sets the value of WDOG\_CNT at initialization, then periodically writes to WDOG\_STAT before the watch dog timer expires. This reloads the watch dog timer with the value from WDOG\_CNT and prevents generation of the expiry event.

If the user does not reload the counter before SCLK\*Count register cycles, an expiry event is generated, and the WDOG\_CTL.WDRO bit is set. When this happens, the counter will stop decrementing and will remain at zero. If the counter is enabled with a zero loaded to the counter, the WDOG\_CTL.WDRO bit is set immediately and the counter remains at zero and does not decrement.

### WDOG\_STAT: Watchdog Timer Status Register - R/W

Reset = 0x0000 0000

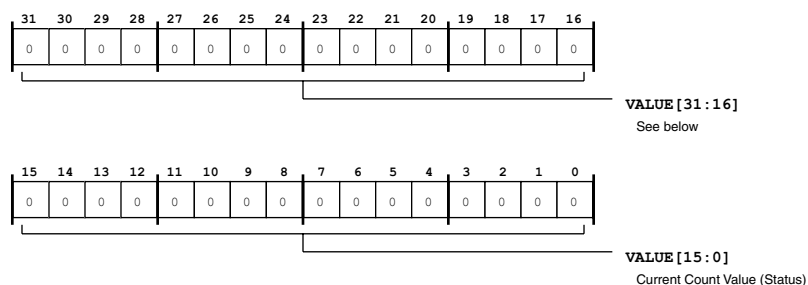


**Figure 16-3:** WDOG\_STAT Register Diagram



**Table 16-6:** WDOG\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 31:0    | VALUE    | Current Count Value (Status). |



# 17 General-Purpose Counter (CNT)

The purpose of this interface is to convert pulses from incremental position encoders into data that is representative of the actual position. This is done by integrating (counting) pulses on one or two inputs. Since integration provides relative position, some devices also feature a zero position input (zero marker) that can be used to establish a reference point to verify that the acquired position does not drift over time. In addition, the incremental position information can be used to determine speed, if the time intervals are measured. The GP counter provides flexible ways to establish position information. When used in conjunction with the GP timer block, the GP counter may allow for the acquisition of coherent position/time-stamp information that enables speed calculation.

## CNT Features

The GP Counter includes the following features:

- 32-bit up/down counter
- Quadrature encode mode (Gray code)
- Binary encoder mode
- Alternative frequency-direction mode
- Timed direction and up/down counting modes
- Zero marker/push button support
- Capture event timing in association with GP Timer
- Boundary comparison and boundary setting features

## CNT Functional Description

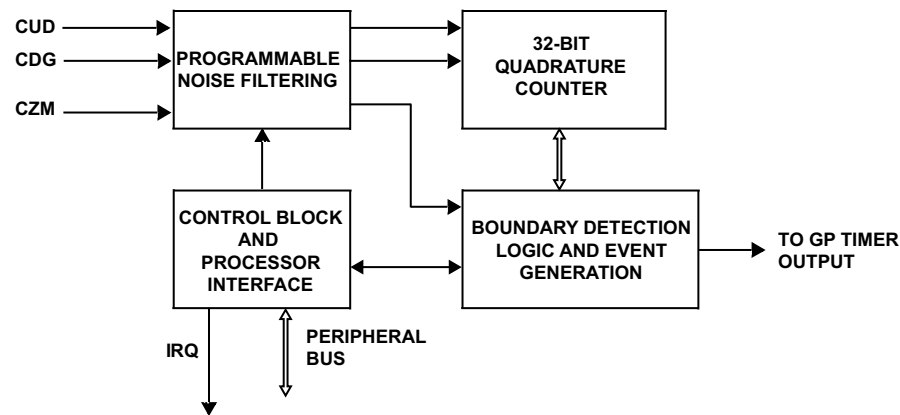
A block diagram of the GP counter is shown below. The CUD and CDG pins accept various forms of incremental inputs and are processed by the 32-bit counter, while the CZM pin can be used to sense the pressing of a push button.

**NOTE:** When enabled, the GP counter requires 3 SCLK cycles of initialization before recognizing valid toggles on its input pins.

The three input pins may be filtered (debounced) before being evaluated by the GP counter.

The GP counter also features a flexible boundary comparison. In all of the operating modes, the counter can be compared to an upper and lower limit. A variety of actions can be taken when these limits are reached.

The module can optionally generate an interrupt request to the system through its IRQ line. On many processors, there is also an output that can be used by a GP Timer module to generate time-stamps on certain events.



CNT Register List

The counter (CNT) provides support for manually controlled rotary controllers, such as the volume wheel on a radio device. This unit also supports industrial encoders.

The CNT converts pulses from incremental position encoders into data that is representative of the actual position. To complete this task, the CNT integrates (counting) pulses on one or two inputs. Because integration provides relative position, some devices also feature a zero position input (zero marker) that establishes a reference point, verifying that the acquired position does not drift over time. The incremental position information may also be used to determine speed, if the time intervals are measured. The CNT provides flexible ways to establish position information. When used in with the , the CNT allows acquisition of coherent position/time-stamp information, enabling speed calculation.

A set of registers govern CNT operations. For more information on CNT functionality, see the CNT register descriptions.

Table 17-1: CNT Module Registers

| Name     | Description             |
|----------|-------------------------|
| CNT_CFG  | Configuration Register  |
| CNT_IMSK | Interrupt Mask Register |

**Table 17-1:** CNT Module Registers (Continued)

| Name       | Description            |
|------------|------------------------|
| CNT_STAT   | Status Register        |
| CNT_CMD    | Command Register       |
| CNT_DEBNCE | Debounce Register      |
| CNT_CNTR   | Counter Register       |
| CNT_MAX    | Maximum Count Register |
| CNT_MIN    | Minimum Count Register |

## ADSP-BF60x CNT Interrupt List

**Table 17-2:** ADSP-BF60x CNT Interrupt List

| Description | ID | DMA Channel | Sensitivity |
|-------------|----|-------------|-------------|
| CNT0 Status | 27 |             | LEVEL       |

## ADSP-BF60x CNT Trigger List

**Table 17-3:** Masters ADSP-BF60x CNT Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
| CNT0 Status | 16 | LEVEL       |
|             |    |             |

**Table 17-4:** Slaves ADSP-BF60x CNT Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

## CNT Operating Modes

The GP counter has the following five modes of operation.

- 1. Quadrature Encoder
- 2. Binary Encoder
- 3. Up/Down Counter
- 4. Direction Counter
- 5. Timed Direction

With the exception of the timed direction mode, the GP counter can operate with the GP timer block to capture additional timing information (time-stamps) associated with events detected by this block.

### Quadrature Encoder Mode

In this mode, the CUD: CDG inputs expect a quadrature-encoded signal that is interpreted as a two-bit Gray code. The order of transitions of the CUD and CDG inputs determines whether the counter increments or decrements. The CNT\_CNTR register contains the number of transitions that have occurred. Refer to the table below for more details. Optionally, an interrupt is generated if both inputs change within one SCLK cycle. Such transitions are not allowed by Gray coding. Therefore, the CNT\_CNTR register remains unchanged, and an error condition is signaled.

**Table 17-5:** Quadrature Events and Counting Mechanism

| CNT_COUNTER<br>Register Value | -4 | -3 | -2 | -1 | 0  | +1 | +2 | +3 | +4 |
|-------------------------------|----|----|----|----|----|----|----|----|----|
| CDG:CUD Inputs                | 00 | 01 | 11 | 10 | 00 | 01 | 11 | 10 | 00 |

It is possible to reverse the count direction of the Gray coded signal. This can be achieved by enabling the polarity inverter of either the CUD pin or the CDG pin. Inverting both pins does not alter the behavior. This feature can be enabled with the CDGINV and CUDINV bits in the CNT\_CFG register.

As an example, if the CDG: CUD inputs are 00 respectively, and the next transition is to 01, this would normally increment the counter as is shown in the table. If the CUD polarity is inverted, this generates a received input of 01 followed by 00. This results in a decrement of the counter, altering the behavior of the connected hardware.

### Binary Encoder Mode

This mode is almost identical to quadrature encoder mode, with the exception that the CUD: CDG inputs expect a binary-encoded signal. The order of transitions of the CUD and CDG inputs determines whether the

counter increments or decrements. The CNT\_CNTR register contains the number of transitions that have occurred. Refer to Table 2. Optionally, an interrupt is generated if the detected code steps by more than 1 (in binary arithmetic) within one SCLK cycle. Such transitions are considered erroneous. Therefore, the CNT\_CNTR register remains unchanged, and an error condition is signaled.

**Table 17-6:** Binary Events and Counting Mechanism

| CNT_COUNTER Register Value | -4 | -3 | -2 | -1 | 0  | +1 | +2 | +3 | +4 |
|----------------------------|----|----|----|----|----|----|----|----|----|
| CDG:CUD Inputs             | 00 | 01 | 10 | 11 | 00 | 01 | 10 | 11 | 00 |

Reversing the CUD and CDG pin polarity has a different effect for the binary encoder mode than for the quadrature encoder mode. Inverting the polarity of the CUD pin only, or inverting both the CUD and CDG pins, results in reversing the count direction.

## Up/Down Counter Mode

In this mode, the counter is incremented or decremented at every active edge of the input pins. If an active edge is detected at the CUD input, the counter increments. The active edge can be selected by the CNT\_CFG.CUDINV bit. If an active edge is detected at the CDG input, the counter decrements. The active edge can be selected by the CNT\_CFG.CDGINV bit. If simultaneous edges occur on CDG and CUD pins, the counter remains unchanged, and both up-count and down-count events are signaled in the CNT\_STAT register.

## Direction Counter Mode

In this mode, the counter is incremented or decremented at every active edge of the CDG input pin. The state of the CUD input determines whether the counter increments or decrements. The polarity can be selected by the CNT\_CFG.CUDINV bit. If an active edge is detected at the CDG input, the counter value changes by one in the selected direction. The active edge can be selected by the CNT\_CFG.CDGINV bit.

## Timed Direction Mode

In this mode, the counter is incremented or decremented at each SCLK cycle. The state of the CUD input determines whether the counter increments or decrements. The polarity can be selected by the CNT\_CFG.CUDINV bit. The CDG pin can be used to gate the clock. The polarity can be selected by the CNT\_CFG.CDGINV bit.

## CNT Event Control

Eleven events can be signaled to the processor using status information and optional interrupt requests. The interrupts are enabled by the respective bits in the CNT\_IMSK register. Dedicated bits in the CNT\_STAT register report events. When an interrupt from the GP counter is acknowledged, the application software

is responsible for correct interpretation of the events. It is recommended to logically AND the content of the `CNT_IMSK` and `CNT_STAT` registers to identify pending interrupts. Interrupt requests are cleared by write-one-to-clear (W1C) operations to the `CNT_STAT` register. Hardware does not clear the status bits automatically, unless the counter module is disabled.

## Illegal Gray/Binary Code Events

When illegal transitions occur in quadrature encoder or binary encoder modes, the `CNT_STAT.ICII` bit is set. If enabled by the `CNT_IMSK.ICIE` bit, an interrupt request is generated. The `ICIE` bit should only be set in the quadrature encoder or binary encoder modes.

## Up/Down Count Events

The `UCII` bit in the `CNT_STAT` register indicates whether the counter has been incremented. Similarly, the `DCII` bit reports decrements. The two events are independent. For instance, if the counter first increments by one and then decrements by two, both bits remain set, even though the resulting counter value shows a decrement by one.

In up/down counter mode, hardware may detect simultaneous active edges on the `CUD` and `CDG` inputs. In that case, the `CNT_CNTR` remains unchanged, but both the `UCII` and `DCII` bits are set. Interrupt requests for these events may be enabled through the `UCIE` and `DCIE` bits. This feature should be used carefully when the counter is clocked at high rates. This is especially critical when the counter operates in `DIR_TMR` mode, as interrupts would be generated every `SCLK` cycle. These events can also be used for additional push buttons, if GP counter features are not needed. When up/down counter mode is enabled, these count events can be used to report interrupts from push buttons that connect to the `CUD` and `CDG` inputs.

## Zero-Count Events

The `CZEROII` status bit indicates that the `CNT_CNTR` has reached a value equal to `0x0000 0000` after an increment or decrement. This bit is not set when the counter value is set to zero by a write to `CNT_CNTR` or by setting the `W1LCNT_ZERO` bit in the `CNT_CMD` register. If enabled by the `CZEROIE` bit, an interrupt request is generated.

## Overflow Events

There are two status bits that indicate whether the signed counter register has overflowed from a positive to a negative value or vice versa. The `COV31II` bit reports that the 32-bit `CNT_COUNT` register has either incremented from `0x7FFF FFFF` to `0x8000 0000`, or decremented from `0x8000 0000` to `0x7FFF FFFF`. If enabled by the `COV31IE` bit, an interrupt request is generated. Similarly, in applications where only the lower 16 bits of the counter are of interest, the `COV15II` status bit reports counter transitions from `0xFFFF 7FFF` to `0xFFFF 8000`, or from `0xFFFF 8000` to `0xFFFF 7FFF`. If enabled by the `COV15IE` bit, an interrupt request is generated.



## Boundary Match Events

The MINCII and MAXCII status bits report boundary events as described in TBD. These bits are not set if the CNT\_CNTR, CNT\_MAX, or CNT\_MIN registers are updated by software or the CNT\_CMD register is written to. The MINCIE and MAXCIE bits in the CNT\_IMSK register enable interrupt generation on boundary events.

## Zero Marker Events

There are three status bits; CZMII, CZMEII and CZMZII associated with zero marker events, as described in TBD. Each of these events can optionally generate an interrupt request, if enabled by the corresponding CZMIE, CZMEIE and CZMZIE bits in the CNT\_IMSK register.

## CNT Programming Model

The following sections provide information used to assist in programming the interface.

### CNT General Programming Flow

The following are general guidelines for configuring and enabling the GP Counter.

1. Initialize (but do not enable) the GP Counter for the desired mode and settings via the CNT\_CFG register.
2. Usually, events of interest are processed using interrupts rather than by polling status bits. If this is the case, clear all status bits and activate the generation of interrupt requests with the CNT\_IMSK register.
3. Configure interrupts at the system level, to insure desired interrupt signalling to the system event controller.
4. If timing information is required, set up the relevant GP Timer in Width Capture mode.
5. Finally, enable interrupts and the GP Counter itself via the CNT\_IMSK and CNT\_CFG registers, respectively.

### CNT Mode Configuration

The CZM input pin can be used to sense the zero marker output of a rotary device or to detect the pressing of a push button. There are four programming schemes, which are functional in all counter modes: push-button mode, zero-marker-zeros-counter mode, zero-marker-error mode, and zero-once mode.

## Configuring GP Counter Push-button Operation

The `CZM` input pin can be used to sense the zero marker output of a rotary device or to detect the pressing of a push button. There are four programming schemes, which are functional in all counter modes: push-button mode, zero-marker-zeros-counter mode, zero-marker-error mode, and zero-once mode.

1. Set `CNT_IMSK.CZME` to enable (unmask) the Zero Marker Error interrupt.
2. Select the active edge polarity through the `CNT_CFG.CZMINV` bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

Now, an active edge at the `CZM` input sets the `CNT_STAT.CZME` bit.

## Configuring Zero-Marker-Zeros-Counter Mode

The following provides information on configuring Zero-Marker-Zeros-Counter mode for the GP Counter.

1. Set `CNT_IMSK.CZMZ` to enable `CNT_CNTR` to be zeroed by a Zero Marker interrupt.
2. Set `CNT_CFG.ZMZC` to enable `ZMZC` mode.
3. Select the active edge polarity through the `CNT_CFG.CZMINV` bit.
4. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

This causes an active level at the `CZM` pin to clear the `CNT_CNTR` register and keep it cleared until the `CZM` pin is deactivated. In addition, the `CNT_STAT.CZMZ` bit is set.

## Configuring Zero-Marker-Error Mode

This mode is used to detect discrepancies between counter value and the zero marker output of certain rotary encoder devices.

1. Set the `CNT_IMSK.CZMEIE` bit to enable this mode.
2. Select the active edge of the `CZM` pin through the `CNT_CFG.CZMINV` bit.
3. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

When an active edge is detected at the `CZM` input pin, the four LSBs of the `CNT_CNTR` register are compared to zero. If they are not zero, a mismatch is signaled by way of the `CNT_STAT.CZMEII` bit.

## Configuring Zero-Once Mode

This mode is used to perform an initial reset of the counter value when an active zero marker is detected. After that, the zero marker is ignored (the counter is not reset anymore).

1. Set the `CNT_CMD.W1ZMONCE` bit to enable this mode.
2. Select the active edge of the `CZM` pin through the `CNT_CFG.CZMINV` bit.
3. Ensure that at least one of the following `CNT_IMSK` bits is enabled: `CZM`, `CZME`, `CZMZ`
4. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

The `CNT_CNTR` register and the `W1ZMONCE` bit are cleared on the next active edge of the `CZM` pin. Now the `W1ZMONCE` bit can be read to check whether the event has already occurred, if desired.

## Configuring Boundary Auto-Extend Mode

In this mode, the boundary registers (`CNT_MIN` and `CNT_MAX`) are modified by hardware whenever the `CNT_CNTR` value reaches either of them. This mode is useful, for example, to keep track of the widest angle a thumbwheel ever reported, even if the software did not generate interrupts.

1. Initialize `CNT_CNTR` with the desired value.
2. Set both `CNT_MIN` and `CNT_MAX` to this same value.
3. Configure the `CNT_CFG.BNDMODE` field for Auto Extend mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

The `CNT_MAX` register is loaded with the current `CNT_CNTR` value if the latter increments beyond the `CNT_MAX` value. Similarly, the `CNT_MIN` register is loaded with the `CNT_CNTR` value if the latter decrements below the `CNT_MIN` value. The `CNT_STAT.MAXCII` and `CNT_STAT.MINCII` status bits are set when the `CNT_CNTR` value matches the respective boundary register value.

## Configuring Boundary Capture Mode

In this mode, the `CNT_CNTR` value is latched into the `CNT_MIN` register at one detected edge of the `CZM` input pin, and latched into `CNT_MAX` at the opposite edge.

1. To capture the `CZM` pin rising edge into `CNT_MIN` and the falling edge into `CNT_MAX`, program `CNT_CFG.CZMINV` for active high polarity. Conversely, to capture the `CZM` pin falling edge into `CNT_MIN` and the rising edge into `CNT_MAX`, program `CNT_CFG.CZMINV` for active low polarity.

2. Program the `CNT_IMSK.MAXCIE` and `CNT_IMSK.MINCIE` interrupt mask bits according to interrupt generation requirements.
3. Configure the `CNT_CFG.BNDMODE` field for Boundary Capture mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

The `CNT_STAT.MAXCII` and `CNT_STAT.MINCII` status bits report the capture event, depending on how interrupt masks are configured.

## Configuring Boundary Compare and Boundary Zero Modes

In these modes, the two boundary registers (`CNT_MAX` and `CNT_MIN`) are simply compared to the value in the `CNT_CNTR` register.

1. Program `CNT_MAX` and `CNT_MIN` registers with appropriate upper and lower range values.
2. Program the `CNT_IMSK.MAXCIE` and `CNT_IMSK.MINCIE` interrupt mask bits according to interrupt generation requirements.
3. Configure the `CNT_CFG.BNDMODE` field for Boundary Compare mode.
4. Proceed with any other desired configuration steps and enable the peripheral.

*RESULT:*

If after incrementing, `CNT_CNTR` equals `CNT_MAX`, then the `CNT_STAT.MAXCII` bit is set. Similarly if after decrementing, `CNT_CNTR` equals `CNT_MIN`, then the `CNT_STAT.MINCII` bit is set. Additionally, for Boundary Zero mode, the counter value in `CNT_CNTR` is set to zero. Note that the `MAXCII` and `MINCII` bits are not set if the `CNT_MAX` and/or `CNT_MIN` registers are updated by software.

## CNT Programming Concepts

Using the features, operating modes, and event control for the GP Counter to their greatest potential requires an understanding of some GP Counter-related concepts. Some key aspects to consider are input noise filtering and capturing timing information.

### CNT Input Noise Filtering

In all modes, the three input pins can be filtered to present clean signals to the GP counter logic. This filtering can be enabled or disabled by the `CNT_CFG.DEBE` bit. The following figure shows the filtering operation for the `CUD` pin.

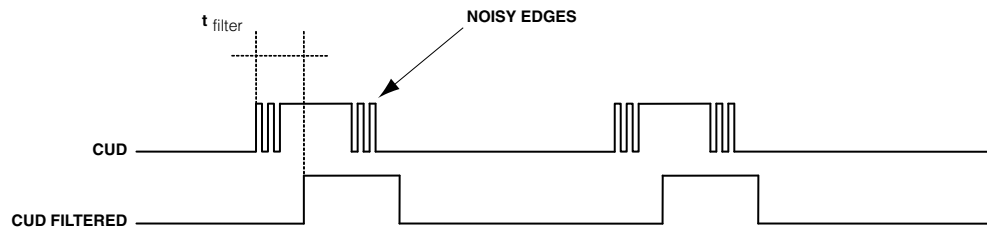


**Figure 17-1:** Programmable Noise Filtering

The filtering mechanism is implemented using counters for each GP counter pin, where each counter is initialized from the `CNT_DEBNCE.DPRESCALE` field. When a transition is detected on a pin, the corresponding counter starts counting up to the programmed number of `SCLK` cycles. The state of the pin is latched after time  $T_{\text{filter}}$  and passed on to the GP counter logic.

The time  $t_{\text{filter}}$  is determined, given `SCLK` and the `CNT_DEBNCE.DPRESCALE` value, by the following formula, where lower values of `DPRESCALE` result in shorter debounce delays:

$$t_{\text{filter}} = 128 \times (2^{\text{DPRESCALE}} \times \text{SCLK})$$

## Capturing Counter Interval and CNT\_CNTR Read Timing

When the count speed is very low, it is often useful to capture the time elapsed since the last count event. In order to do this, program the associated GP Timer's `TMR_CFG` register in a Width Capture mode with `PULSEHI = 0`, `TMODE=b#1011` and `TINSEL = 1`.

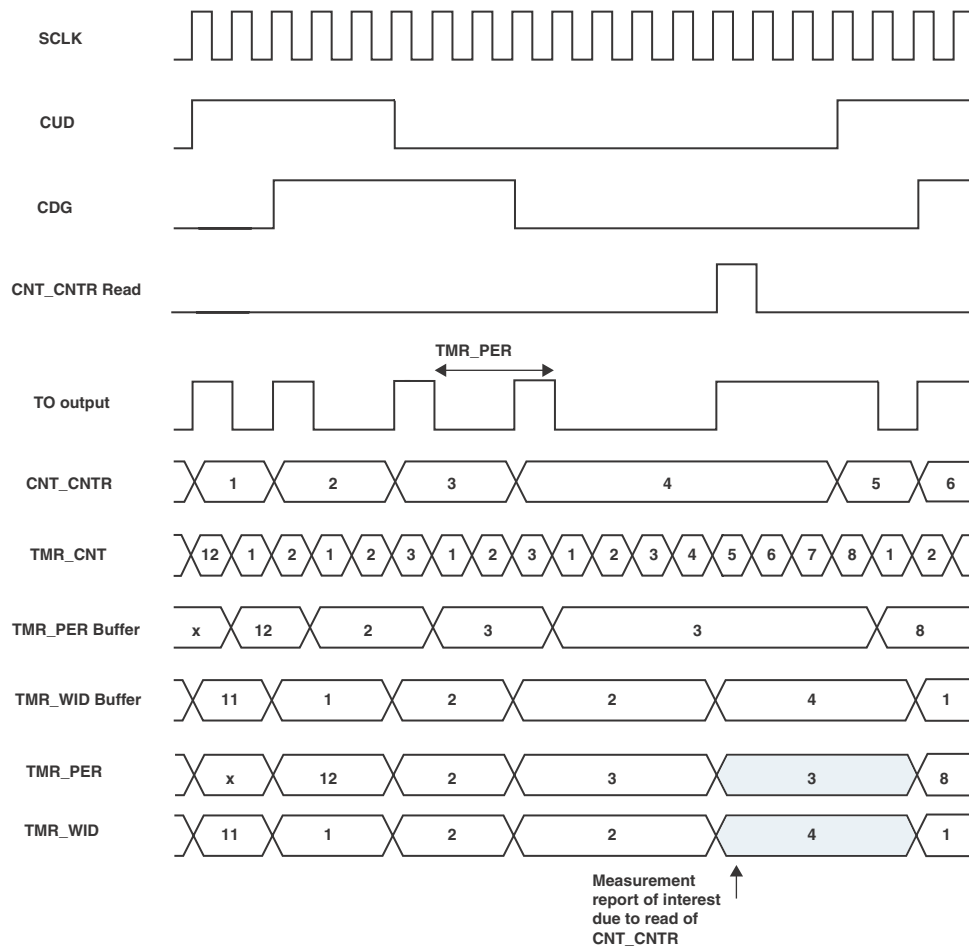


**Figure 17-2:** Capture Intervals

The following figure shows the operation of the GP counter and the GP timer in this mode.

The TO signal generates a pulse every time a count event occurs. In addition, when the processor reads the CNT\_CNTR register, the TO signal presents a pulse which is extended (high) until the next count event. The GP timer updates its TMR\_PER register with the period (measured from falling edge to falling edge, because PULSEHI = 0) of the TO signal. The TMR\_WID register is updated with the pulse width (the portion where TO is low, again because PULSEHI = 0). Both registers are updated at every rising edge of the TO signal (because TMR\_CFG.TMODE=b#011).

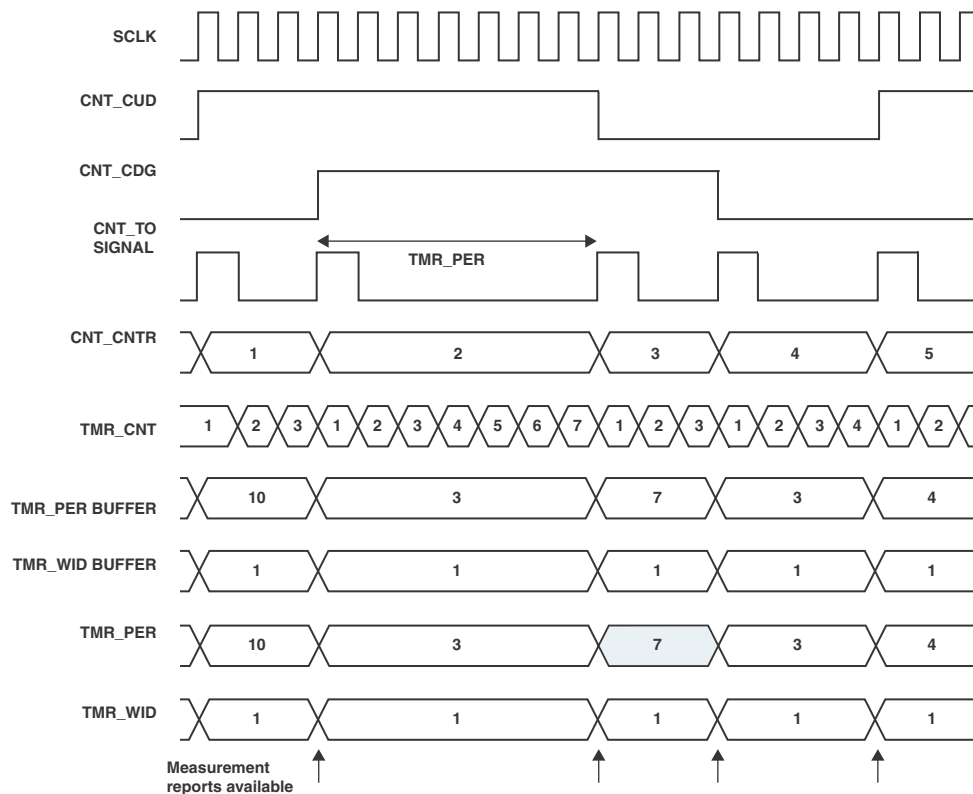
Therefore, the TMR\_PER register contains the period between the last two count events, and the TMR\_WID register contains the time since the last count event and the read of the CNT\_CNTR register, both measured in SCLK cycles.

Read the CNT\_CNTR register to latch the two time measurements, providing a coherent triplet of information to calculate speed and position.

**NOTE:** Speed restrictions apply to the use of the T0 signal. Therefore, the user must take care to not operate at very high count event rates. For instance, if CNT\_CNTR is incremented/decremented every SCLK cycle (timed direction mode), the T0 signal will not be valid.

## Capturing Time Interval Between Successive Counter Events

When the only timing information of interest is the interval between successive count events, the associated timer should be programmed in a width capture mode with TMR\_CFG bit settings of PULSEHI = 1, TMODE=b#1010 and TINSEL = 1. Typically, this information is sufficient if the speed of GP counter events is known not to reach very low values.



### Figure 17-3: Period Register Timing

The figure above shows the operation of the GP Counter and the GP timer in this mode. The T0 signal generates a pulse every time a count event occurs. The GP timer updates its TMR\_PER register with the period (measured from rising edge to rising edge) of the T0 signal. The TMR\_PER register is updated at every rising edge of the T0 signal and contains the number of SCLK cycles that have elapsed since the previous rising edge.

Incidentally, the TMR\_WID register is also updated at the same time, but is generally of no interest in this mode of operation. If no reads of the CNT\_CNTR register occur between counter events, the TMRn\_WID register only contains the width of the T0 pulse. If a read of CNT\_CNTR has occurred between events, the TMR\_WID register contains the time between the read of CNT\_CNTR and the next event.

This mode can also be used with `PULSEHI = 0`. In this case, the period of `T0` is measured between falling edges. It results in the same values as in the previous case, only the latching occurs one `SCLK` cycle later.

## CNT Register Descriptions

CNT (CNT) contains the following registers.

**Table 17-7:** CNT Module Registers

| Name       | Description             |
|------------|-------------------------|
| CNT_CFG    | Configuration Register  |
| CNT_IMSK   | Interrupt Mask Register |
| CNT_STAT   | Status Register         |
| CNT_CMD    | Command Register        |
| CNT_DEBNCE | Debounce Register       |
| CNT_CNTR   | Counter Register        |
| CNT_MAX    | Maximum Count Register  |
| CNT_MIN    | Minimum Count Register  |

### Configuration Register

The `CNT_CFG` register configures counter modes, configures input pins, and enable the CNT.



**CNT\_CFG: Configuration Register - R/W**

Reset = 0x0000

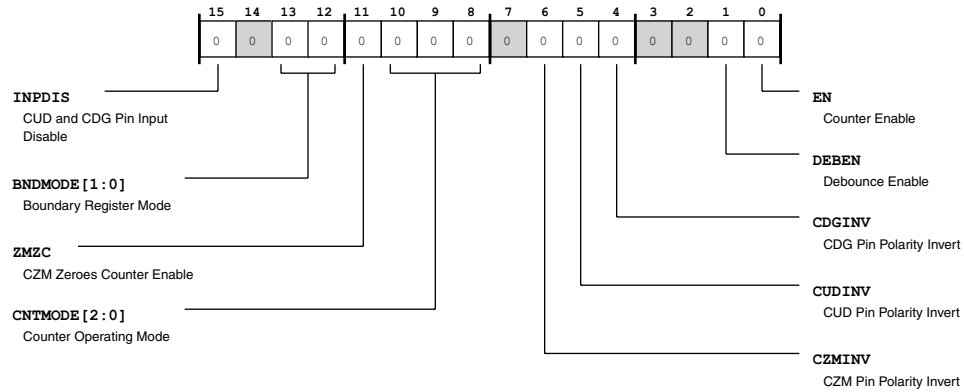


**Figure 17-4:** CNT\_CFG Register Diagram

**Table 17-8:** CNT\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------|
| 15      | INPDIS   | CUD and CDG Pin Input Disable.<br>The CNT_CFG.INPDIS disables or enables the CNT_UD input pin and the CNT_DG pin. |
|         |          | 0      Enable                                                                                                     |
|         |          | 1      Pin Input Disable                                                                                          |
| 13:12   | BNDMODE  | Boundary Register Mode.<br>The CNT_CFG.BNDMODE selects the mode for the CNT_MIN and CNT_MAX boundary registers.   |
|         |          | 0      BND_COMP<br>Boundary compare mode                                                                          |
|         |          | 1      BIN_ENC<br>Binary encoder mode                                                                             |
|         |          | 2      BND_CAPT<br>Boundary capture mode                                                                          |
|         |          | 3      BND_AEXT<br>Boundary auto-extend mode                                                                      |

**Table 17-8:** CNT\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | ZMZC     | CZM Zeroes Counter Enable.<br>The CNT_CFG.ZMZC enables or disables level sensitive - Active CNT_ZM pin operation to zero the CNT_CNTR.                                                                              |
|         |          | 0      Disable                                                                                                                                                                                                      |
|         |          | 1      Enable                                                                                                                                                                                                       |
| 10:8    | CNTMODE  | Counter Operating Mode.<br>The CNT_CFG.CNTMODE selects the operating mode for the CNT_UD input pin and the CNT_DG pin.                                                                                              |
|         |          | 0      QUAD_ENC<br>Quadrature encoder mode                                                                                                                                                                          |
|         |          | 1      BIN_ENC<br>Binary encoder mode                                                                                                                                                                               |
|         |          | 2      UD_CNT<br>Rotary counter mode                                                                                                                                                                                |
|         |          | 4      DIR_CNT<br>Direction counter mode                                                                                                                                                                            |
|         |          | 5      DIR_TMR<br>Direction timer mode                                                                                                                                                                              |
| 6       | CZMINV   | CZM Pin Polarity Invert.<br>The CNT_CFG.CZMINV selects the polarity for the CNT_ZM pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled. |
|         |          | 0      Active High, Rising Edge                                                                                                                                                                                     |
|         |          | 1      Active Low, Falling Edge                                                                                                                                                                                     |

**Table 17-8:** CNT\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | CUDINV   | CUD Pin Polarity Invert.<br>The CNT_CFG.CUDINV selects the polarity for the CNT_UD pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled. |
|         |          | 0      Active High, Rising Edge                                                                                                                                                                                     |
|         |          | 1      Active Low, Falling Edge                                                                                                                                                                                     |
| 4       | CDGINV   | CDG Pin Polarity Invert.<br>The CNT_CFG.CDGINV selects the polarity for the CNT_DG pin. This polarity must be configured before the counter is enabled. It must not change on-the-fly while the counter is enabled. |
|         |          | 0      Active High, Rising Edge                                                                                                                                                                                     |
|         |          | 1      Active Low, Falling Edge                                                                                                                                                                                     |
| 1       | DEBEN    | Debounce Enable.<br>The CNT_CFG.DEBEN enables or disables CNT input debounce filtering operation selected with the CNT_DEBNCE register.                                                                             |
|         |          | 0      Disable                                                                                                                                                                                                      |
|         |          | 1      Enable                                                                                                                                                                                                       |
| 0       | EN       | Counter Enable.<br>The CNT_CFG.EN enables or disables CNT operation.                                                                                                                                                |
|         |          | 0      Counter Disable                                                                                                                                                                                              |
|         |          | 1      Counter Enable                                                                                                                                                                                               |

## Interrupt Mask Register

The CNT\_IMSK register supports enabling (unmasking) interrupt request generation from each of the CNT events.

All bits in CNT\_IMSK either disable/mask an interrupt (if bit cleared) or enable/unmask an interrupt (if bit set).

CNT\_IMSK: Interrupt Mask Register - R/W

Reset = 0x0000

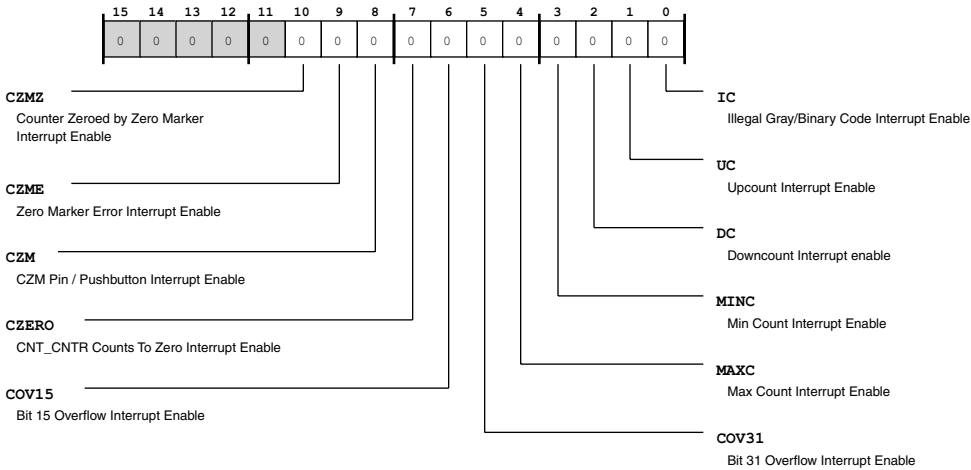


Figure 17-5: CNT\_IMSK Register Diagram

Table 17-9: CNT\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                         |
|---------|----------|-------------------------------------------------|
| 10      | CZMZ     | Counter Zeroed by Zero Marker Interrupt Enable. |
|         |          | 0 Mask Interrupt                                |
|         |          | 1 Unmask Interrupt                              |
| 9       | CZME     | Zero Marker Error Interrupt Enable.             |
|         |          | 0 Mask Interrupt                                |
|         |          | 1 Unmask Interrupt                              |
| 8       | CZM      | CZM Pin / Pushbutton Interrupt Enable.          |
|         |          | 0 Mask Interrupt                                |
|         |          | 1 Unmask Interrupt                              |
| 7       | CZERO    | CNT_CNTR Counts To Zero Interrupt Enable.       |
|         |          | 0 Mask Interrupt                                |
|         |          | 1 Unmask Interrupt                              |

**Table 17-9:** CNT\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                    |                  |
|---------|----------|--------------------------------------------|------------------|
| 6       | COV15    | Bit 15 Overflow Interrupt Enable.          |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 5       | COV31    | Bit 31 Overflow Interrupt Enable.          |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 4       | MAXC     | Max Count Interrupt Enable.                |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 3       | MINC     | Min Count Interrupt Enable.                |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 2       | DC       | Downcount Interrupt enable.                |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 1       | UC       | Upcount Interrupt Enable.                  |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |
| 0       | IC       | Illegal Gray/Binary Code Interrupt Enable. |                  |
|         |          | 0                                          | Mask Interrupt   |
|         |          | 1                                          | Unmask Interrupt |

## Status Register

The CNT\_STAT register provides status information for each of the CNT events. When a CNT event is detected, the corresponding bit in this register is set. It remains set until either software writes a "1" to the bit (write-1-to-clear) or the CNT is disabled.

All bits in CNT\_STAT indicate either no interrupt pending (if bit cleared) or an interrupt pending (if bit set).

**CNT\_STAT: Status Register - R/W**

Reset = 0x0000

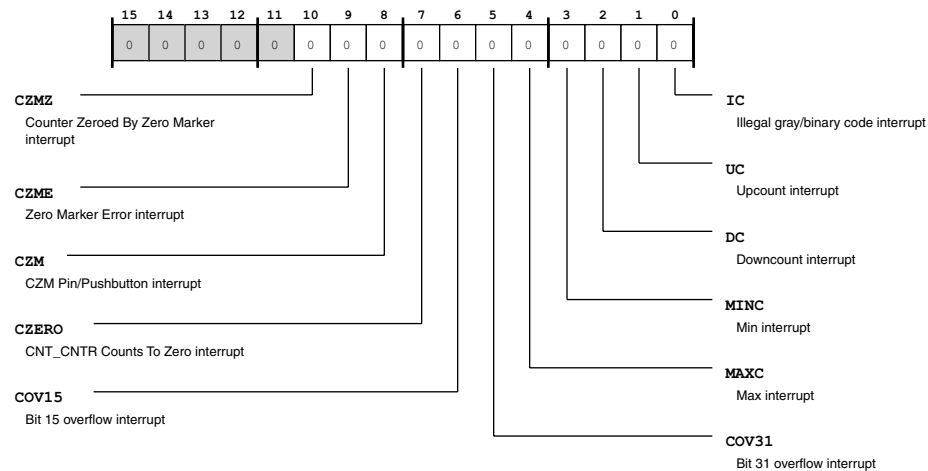


**Figure 17-6:** CNT\_STAT Register Diagram

**Table 17-10:** CNT\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                  |
|---------|----------|------------------------------------------|
| 10      | CZMZ     | Counter Zeroed By Zero Marker interrupt. |
| 9       | CZME     | Zero Marker Error interrupt.             |
| 8       | CZM      | CZM Pin/Pushbutton interrupt.            |
| 7       | CZERO    | CNT_CNTR Counts To Zero interrupt.       |
| 6       | COV15    | Bit 15 overflow interrupt.               |
| 5       | COV31    | Bit 31 overflow interrupt.               |
| 4       | MAXC     | Max interrupt.                           |
| 3       | MINC     | Min interrupt.                           |

**Table 17-10:** CNT\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration             |
|---------|----------|-------------------------------------|
| 2       | DC       | Downcount interrupt.                |
| 1       | UC       | Upcount interrupt.                  |
| 0       | IC       | Illegal gray/binary code interrupt. |

## Command Register

The `CNT_CMD` register configures the CNT, enabling operations such as zeroing a counter register and copying or swapping boundary registers. These actions are taken by setting the appropriate bit.

Read operations from this register do not return meaningful values, with the exception of the `CNT_CMD.W1ZMONCE` bit, where a set bit indicates that:

the bit has been set by software before, but

no zero marker event has been detected on the `CNT_ZM` pin yet.

For more information, see the CNT functional description.

The `CNT_CNTR`, `CNT_MIN`, and `CNT_MAX` registers can be initialized to zero by setting the `CNT_CMD.W1LCNTZERO`, `CNT_CMD.W1LMINZERO`, and `CNT_CMD.W1LMAXZERO` bits. In addition to clearing registers, `CNT_CMD` permits modifying the `CNT_MIN` and `CNT_MAX` boundary registers in a number of ways. The current counter value in `CNT_CNTR` can be captured and loaded into either of the two boundary registers to create new boundary limits. This operation is performed by setting the `CNT_CMD.W1LMAXCNT` and `CNT_CMD.W1LMINCNT` bits. Alternatively, the counter can be loaded from `CNT_MAX` or `CNT_MIN` using the `CNT_CMD.W1LCNTMAX` and `CNT_CMD.W1LCNTMIN` bits. It is also possible to transfer the current `CNT_MAX` value into `CNT_MIN` (or vice versa) through the `CNT_CMD.W1LMINMAX` and `CNT_CMD.W1LMAXMIN` bits.

Another counter operation is the ability to only have the zero marker clear the `CNT_CNTR` register once. For more information, see the CNT functional description.

It is possible for multiple actions to be performed simultaneously by setting multiple bits in the `CNT_CMD` register, but there are restrictions. The bits associated with each command have been grouped together such that all bits that involve a write to the `CNT_CNTR`, `CNT_MAX`, or `CNT_MIN` are located within bits 4-bit groups of the `CNT_CMD` register.

Note that a maximum of three commands can be issued at any one time, excluding the `CNT_CMD.W1ZMONCE` command. Also note that `CNT_CMD.W1LCNTMIN`, `CNT_CMD.W1LCNTMAX`, and `CNT_CMD.W1LCNTZERO` bits have to be used exclusively. Never set more than one of them at the same time.

**CNT\_CMD: Command Register - R/WA**

Reset = 0x0000

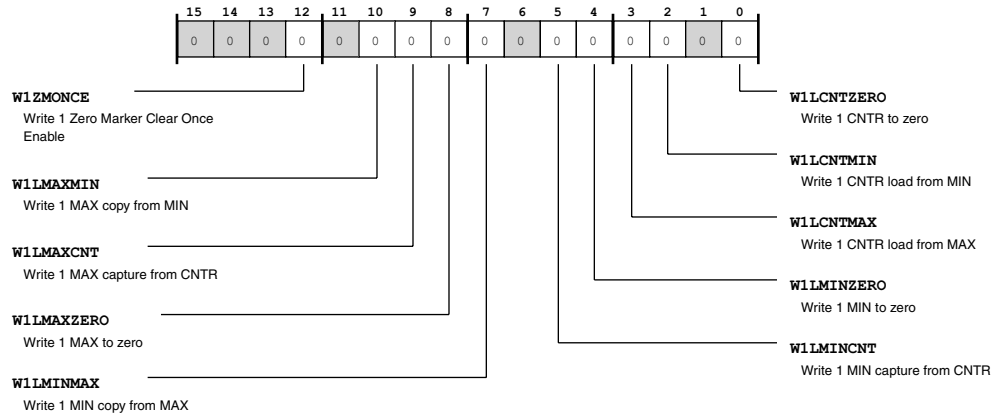


**Figure 17-7: CNT\_CMD Register Diagram**

**Table 17-11: CNT\_CMD Register Fields**

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                            |
|---------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | W1ZMONCE   | Write 1 Zero Marker Clear Once Enable.<br>The CNT_CMD.W1ZMONCE enables a single zero marker clear of CNT_CNTR. Reading a "1" in this bit indicates that the bit has been set by software before, but no zero marker event has been detected on the CNT_ZM pin yet. |
| 10      | W1LMAXMIN  | Write 1 MAX copy from MIN.                                                                                                                                                                                                                                         |
| 9       | W1LMAXCNT  | Write 1 MAX capture from CNTR.                                                                                                                                                                                                                                     |
| 8       | W1LMAXZERO | Write 1 MAX to zero.                                                                                                                                                                                                                                               |
| 7       | W1LMINMAX  | Write 1 MIN copy from MAX.                                                                                                                                                                                                                                         |
| 5       | W1LMINCNT  | Write 1 MIN capture from CNTR.                                                                                                                                                                                                                                     |
| 4       | W1LMINZERO | Write 1 MIN to zero.                                                                                                                                                                                                                                               |
| 3       | W1LCNTMAX  | Write 1 CNTR load from MAX.                                                                                                                                                                                                                                        |
| 2       | W1LCNTMIN  | Write 1 CNTR load from MIN.                                                                                                                                                                                                                                        |
| 0       | W1LCNTZERO | Write 1 CNTR to zero.                                                                                                                                                                                                                                              |



## Debounce Register

The CNT\_DEBNCE register selects the noise filtering characteristic of the three input pins according to the formula:

$$t_{\text{filter}} = 128 \times (2^{\text{DPRESCALE}} / \text{SCLK})$$

**CNT\_DEBNCE: Debounce Register - R/W**

Reset = 0x0000

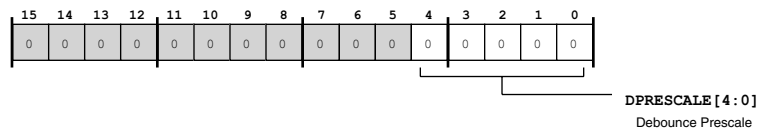


**Figure 17-8:** CNT\_DEBNCE Register Diagram



**Table 17-12:** CNT\_DEBNCE Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                        |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4:0     | DPRESCALE | Debounce Prescale.<br>The CNT_DEBNCE.DPRESCALE selects the desired number of input filtering cycles ( and resulting input debounce time) in multiples of SCLK. |
|         |           | 0                      1x cycles = 128 SCLK cycles                                                                                                             |
|         |           | 1                      2x cycles                                                                                                                               |
|         |           | 2                      4x cycles                                                                                                                               |
|         |           | 3                      8x cycles                                                                                                                               |
|         |           | 4                      16x cycles                                                                                                                              |
|         |           | 5                      32x cycles                                                                                                                              |
|         |           | 6                      64x cycles                                                                                                                              |
|         |           | 7                      128x cycles                                                                                                                             |
|         |           | 8                      256x cycles                                                                                                                             |
|         |           | 9                      512x cycles                                                                                                                             |
|         |           | 10                     1024x cycles                                                                                                                            |
|         |           | 11                     2048x cycles                                                                                                                            |
|         |           | 12                     4096x cycles                                                                                                                            |
|         |           | 13                     8192x cycles                                                                                                                            |
|         |           | 14                     16384x cycles                                                                                                                           |
|         |           | 15                     32768x cycles                                                                                                                           |
|         |           | 16                     65536x cycles                                                                                                                           |
|         |           | 17                     131072x cycles                                                                                                                          |
|         |           | 18                     Reserved from this value<br>10010 - 11111: Reserved                                                                                     |
|         |           | 31                     Reserved till this value                                                                                                                |

## Counter Register

The CNT\_CNTR register holds the 32-bit, twos-complement count value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows use of the CNT as a 16-bit counter if sufficient for the application.

CNT\_CNTR: Counter Register - R/W

Reset = 0x0000 0000

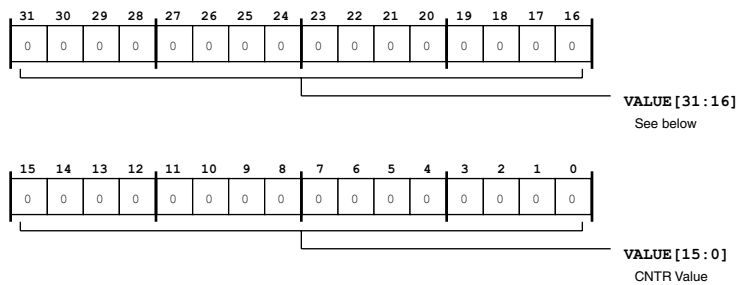


Figure 17-9: CNT\_CNTR Register Diagram

Table 17-13: CNT\_CNTR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | CNTR Value.             |

## Maximum Count Register

The CNT\_MAX register holds the 32-bit, twos-complement, higher boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

**CNT\_MAX: Maximum Count Register - R/W**

Reset = 0x0000 0000

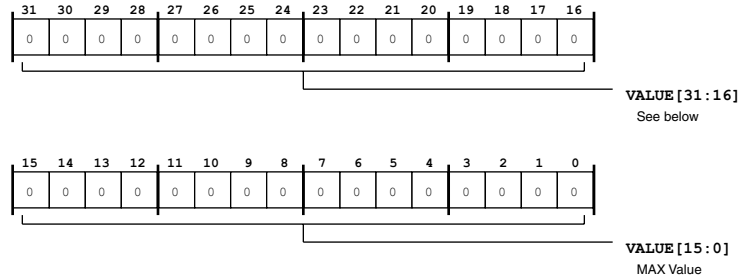


**Figure 17-10:** CNT\_MAX Register Diagram

**Table 17-14:** CNT\_MAX Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | MAX Value.              |

## Minimum Count Register

The CNT\_MIN register holds the 32-bit, twos-complement, lower boundary value. It can be read and written at any time. Hardware ensures that reads and write are atomic, by providing respective shadow registers. This register can be accessed with either 32-bit or 16-bit operations. This allows for using the CNT as a 16-bit counter if sufficient for the application.

**CNT\_MIN: Minimum Count Register - R/W**

Reset = 0x0000 0000

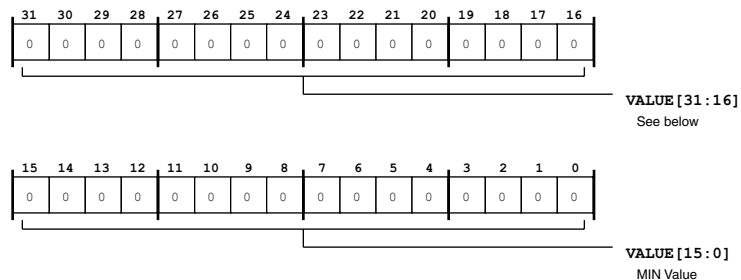


**Figure 17-11:** CNT\_MIN Register Diagram

**Table 17-15:** CNT\_MIN Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | MIN Value.              |

# 18 Pulse-Width Modulator (PWM)

The Pulse Width Modulator (PWM) module is a flexible and programmable waveform generator. With minimal CPU intervention the PWM peripheral is capable of generating complex waveforms for motor control, Pulse Coded Modulation (PCM), Digital to Analog Conversion (DAC) functionality, power switching and power conversion. The PWM module has 4 PWM pairs capable of 3-phase PWM generation for source inverters for AC induction and DC brush less motors.

## PWM Features

The two 3-phase PWM generation units each feature:

- 16-bit center-based PWM generation unit
- Programmable PWM pulse width
- Single/double update modes
- Programmable dead time and switching frequency
- Twos-complement implementation which permits smooth transition to full ON and full OFF states
- Dedicated asynchronous PWM shutdown signal

## PWM Register List

The pulse-width modulator (PWM) includes multiple timers (providing period flexibility) and channels (provide mode, interrupt, and pulse shape flexibility), permitting a wide variety of PWM output options for motor control and other applications. A set of registers govern PWM operations. For more information on PWM functionality, see the PWM register descriptions.

**Table 18-1:** PWM Module Registers

| Name        | Description             |
|-------------|-------------------------|
| PWM_CTL     | Control Register        |
| PWM_CHANCFG | Channel Config Register |
| PWM_TRIPCFG | Trip Config Register    |

**Table 18-1:** PWM Module Registers (Continued)

| Name         | Description                    |
|--------------|--------------------------------|
| PWM_STAT     | Status Register                |
| PWM_IMSK     | Interrupt Mask Register        |
| PWM_ILAT     | Interrupt Latch Register       |
| PWM_CHOPCFG  | Chop Configuration Register    |
| PWM_DT       | Dead Time Register             |
| PWM_SYNC_WID | Sync Pulse Width Register      |
| PWM_TM0      | Timer 0 Period Register        |
| PWM_TM1      | Timer 1 Period Register        |
| PWM_TM2      | Timer 2 Period Register        |
| PWM_TM3      | Timer 3 Period Register        |
| PWM_TM4      | Timer 4 Period Register        |
| PWM_DLYA     | Channel A Delay Register       |
| PWM_DLYB     | Channel B Delay Register       |
| PWM_DLYC     | Channel C Delay Register       |
| PWM_DLYD     | Channel D Delay Register       |
| PWM_ACTL     | Channel A Control Register     |
| PWM_AH0      | Channel A-High Duty-0 Register |
| PWM_AH1      | Channel A-High Duty-1 Register |
| PWM_AL0      | Channel A-Low Duty-0 Register  |
| PWM_AL1      | Channel A-Low Duty-1 Register  |
| PWM_BCTL     | Channel B Control Register     |
| PWM_BH0      | Channel B-High Duty-0 Register |



**Table 18-1:** PWM Module Registers (Continued)

| Name     | Description                          |
|----------|--------------------------------------|
| PWM_BH1  | Channel B-High Duty-1 Register       |
| PWM_BL0  | Channel B-Low Duty-0 Register        |
| PWM_BL1  | Channel B-Low Duty-1 Register        |
| PWM_CCTL | Channel C Control Register           |
| PWM_CH0  | Channel C-High Pulse Duty Register 0 |
| PWM_CH1  | Channel C-High Pulse Duty Register 1 |
| PWM_CL0  | Channel C-Low Pulse Duty Register 0  |
| PWM_CL1  | Channel C-Low Duty-1 Register        |
| PWM_DCTL | Channel D Control Register           |
| PWM_DH0  | Channel D-High Duty-0 Register       |
| PWM_DH1  | Channel D-High Pulse Duty Register 1 |
| PWM_DL0  | Channel D-Low Pulse Duty Register 0  |
| PWM_DL1  | Channel D-Low Pulse Duty Register 1  |

## ADSP-BF60x PWM Interrupt List

**Table 18-2:** ADSP-BF60x PWM Interrupt List

| Description       | ID | DMA Channel | Sensitivity |
|-------------------|----|-------------|-------------|
| PWM0 PWMTMR Group | 28 |             | LEVEL       |
| PWM0 Trip         | 29 |             | LEVEL       |
| PWM1 PWMTMR Group | 30 |             | LEVEL       |
| PWM1 Trip         | 31 |             | LEVEL       |

## ADSP-BF60x PWM Trigger List

**Table 18-3:** Masters ADSP-BF60x PWM Trigger List

| Description       | ID | Sensitivity |
|-------------------|----|-------------|
| PWM0 PWMTMR Group | 17 | LEVEL       |
| PWM1 PWMTMR Group | 18 | LEVEL       |
|                   |    |             |

**Table 18-4:** Slaves ADSP-BF60x PWM Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

## Functional Description

Each PWM Module has 4 PWM pairs with 2 outputs: a high and a low side signal.

During initialization, the `PWM_TMR` register is written to define the PWM period, and the Channel Pulse Duty registers are written to define the initial channel pulse widths. The `PWM_CTL` and `PWM_CHANCFG` registers are written, depending on the system configuration and modes. The `PWM_STAT` register can be read to determine polarity, and whether switched reluctance (SR) mode (`PWM_SR` bit) is enabled, and whether an external trip situation is preventing the correct start-up of the PWM Controller. An active external trip event must be resolved prior to PWM startup. The `PWM_CTL` register is then written to define the major operating mode and to enable the PWM outputs and PWM sync pulse.

During the `PWM_SYNCINT` interrupt driven control loop, only the `PWM_AH0.DUTY` values are updated typically.

Typically the `PWMx_TMR` interrupt is used to periodically execute an Interrupt Service Routine (ISR), to update the PWM channel control and duty registers according to a control algorithm based on expected operation and sampled existing operation. The `PWMx_TMR` interrupt also can trigger an ADC to sample data for use during the ISR.

During processor boot the PWM is initialized and the program flow enters a wait loop. When a `PWM_SYNC` interrupt occurs, the ADC samples data, the data is algorithmically interpreted, and new PWM channel duties are calculated and written to the PWM. More sophisticated implementations include

different startup, runtime, and shutdown algorithms to determine PWM channel duties based on expected behavior and further features.

During initialization, the `PWM_TMO` registers are written to define the PWM periods of different timers. The `PWM_DLYA` registers are written to define the initial phase difference between the main timer and the other timers. The `PWM_CHANCFG` and `PWM_ACTL` registers are written to define the initial pulse configuration. The Channel x High/Low Duty registers are programmed to define the initial duty values. The requisite dead-time must be written into the `PWM_DT` register. The `PWM_CTL` register is written in the end with the final configurations and enable bit for the entire PWM (`PWM_CTL.GLOBEN`).

The `PWM_SYNC` interrupt is assigned to one of the core's user interrupts. During the `PWM_SYNC` interrupt driven control loop, only the `PWM_DLYA`, the duty registers and channel C high pulse duty register values are typically updated. To see programming limitations on the PWM registers, see the Register Descriptions section.

During any external trip event (if not disabled), the PWM outputs will be turned off. When a PWM output is turned off, it means that the output level is held at a polarity opposite that given in the `PWM_CHANCFG.POLDH` bits in the `PWM_CHANCFG` register. The PWM sync pulse will continue to operate if it is already enabled. A `PWMTRIP` interrupt occurs if unmasked, to notify the software of this event.

Note that even if the clock to the PWM is damaged, an external trip event will turn off the PWM outputs, but the `PWMTRIP` interrupt may not occur.

## Architectural Concepts

The PWM Controller is driven by a clock, whose period is  $t_{SCLK}$ . The PWM generator produces four pairs (four high-side and four low-side) of PWM signals on the eight PWM output pins. Each high and low pair constitutes a channel. For example `PWM_AL/PWM_AH` make up channel A, and `PWM_BLPWM_BH` make up channel B and so on. Each pair of channel outputs can be produced with reference to either a main timer or to an independent timer. These timers operate on a switching frequency determined by the `PWM_TMO` registers. There are 2 duty registers for every PWM output, which enable generation of symmetrical or asymmetrical waveforms that produce lower harmonic distortion in three-phase PWM inverters, with minimal CPU intervention.

## Block Diagram

The following figure shows a block diagram that represents the main functional blocks of the PWM controller.

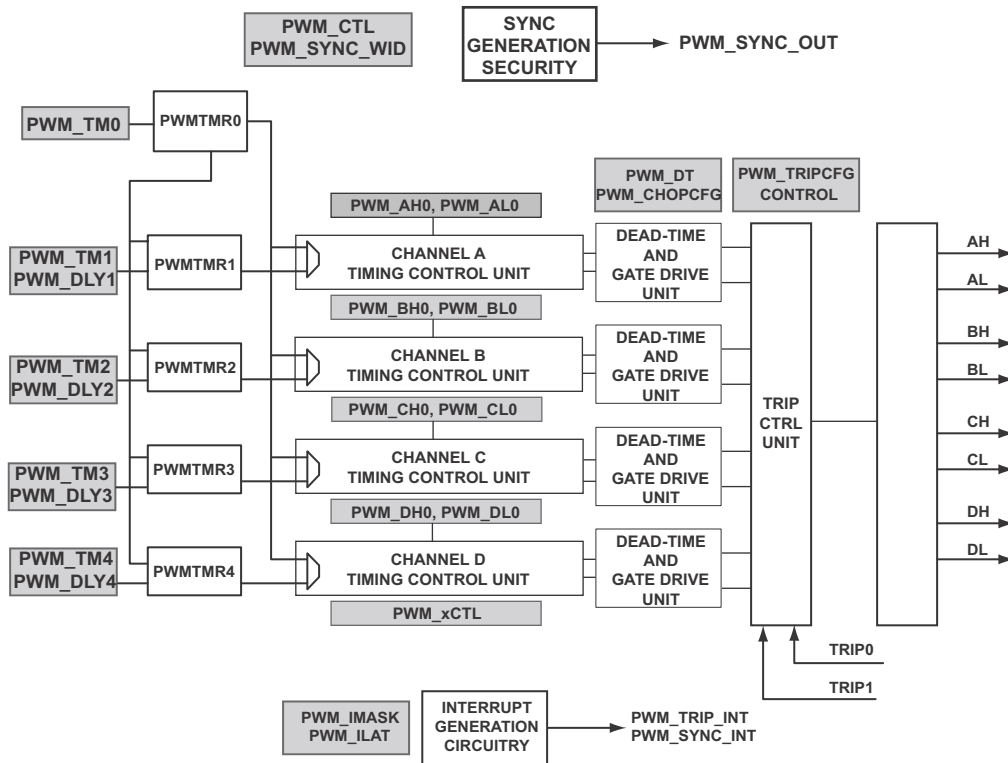


**Figure 18-1: PWM Block Diagram**

The primary blocks are described below.

- Each pair of PWM signals is referenced either to the main timer or to the independent timer.
- PWMTMR0 is the main timer and can trigger the delayed start of the other timers.
- Timing Control Units, one for each channel, which together form the core of the PWM, generate the required complex waveforms on the high side and low side outputs for the respective channel.
- Dead Time insertion is done after the ideal PWM output pair is generated.
- The Gate Drive Unit generates the high-frequency chopping signal and subsequently mixes it with the requisite PWM output signals.
- The PWM Shutdown and Interrupt Controller manages the various PWM shutdown modes for the timing unit and generates the requisite interrupt signals.
- The PWM Sync Pulse Control Unit generates the internal PWM\_SYNC pulse and also controls whether the external PWM\_SYNC input pulse is used.

## Timer Units

Five timers make up the time base for the PWM module. The main timer, PWMTMR0 operates at a switching frequency determined by the period register PWM\_TM0. The four remaining timers PWMTMR1,

PWMTMR2, PWMTMR3 and PWMTMR4 can operate at independent switching frequencies determined by their respective registers.

These four timers can be programmed to work at a multiple of the main timer's frequency by programming respective PWM\_TMx appropriately. In this case, the PWM\_DLYA registers can be used to provide for lead-lag phase control of a given timer with respect to the main timer PWMTMR0.

**NOTE:** When delayed operation of a timer is enabled, its register value must either be equal to the PWM\_TMO register value or PWM\_TMO must be an integer multiple of each timer's register to ensure proper function. Non-integer multiples are not allowed.

## PWM Switching Frequency (PWM\_TM) Register

The 16-bit read/write PWM period register controls the PWM switching frequency. The fundamental timing unit of the PWM Controller is  $t_{SCLK}$ . Therefore, for a 100 MHz system clock (SCLK),  $f_{SCLK}$ , the fundamental time increment ( $t_{SCLK}$ ) is 10 ns. The value written to a timer's register is effectively the number of  $t_{SCLK}$  clock increments in half a PWM period. The required timer register value as a function of the desired PWM switching frequency ( $f_{PWM}$ ) is given by:

$$PWM\_TM = f_{SCLK} / 2 \times f_{PWM}$$

Therefore, the PWM switching period ( $T_s$ ) can be written as:

$$T_s = 2 \times PWM\_TM \times t_{SCLK}$$

For example, for an  $f_{SCLK}$  of 100 MHz and a desired PWM switching frequency ( $f_{PWM}$ ) of 10 kHz ( $T_s = 100$  ms), the correct value to load into the timer register is:

$$PWM\_TM = 100 \times 10^6 \div 2 \times 10 \times 10^3 = 5000$$

The largest value that can be written to the 16-bit timer register is 0xFFFF = 65,535, which, at an  $f_{SCLK}$  of 100 MHz, corresponds to a minimum PWM switching frequency of:

$$f_{PWM(min)} = 100 \times 10^6 \div 2 \times 65535 = 762 \text{ Hz}$$

**NOTE:** Timer register values of 0 and 1 are not defined and should not be used when the PWM outputs or PWM is enabled.

## Timer Unit Operation

The PWM timer is an up-down counter operates on the peripheral clock of period  $t_{CK}$ . The period of the PWM timer is divided into two halves. In the first half, the timer roughly counts down from  $PWM\_TMx/2$  to  $-PWM\_TMx/2$ . During this half, the PWM\_STAT.TMR0PHASE register is held at 0. In the second half of the period, the timer roughly counts up from  $-PWM\_TMx/2$  to  $PWM\_TMx/2$ . The PWM\_STAT.TMR0PHASE bit indicates a 1 during this half.

The actual partition of the periods varies slightly between odd and even values of the half-period, PWM\_TMx.

When the timer register value is odd, for example 11, then that timer loads +5 at the beginning of the period, counts down from +5 to -5 in the first half, reloads -5 at the midpoint and counts up from -5 to +5 in the second half. The reload values at the period and mid-period boundaries are the same as the previous count. It totally counts  $2 \times 11 = 22$  counts in the entire period. This is shown in the figure below. Note that both half-periods have a count of 11 each.

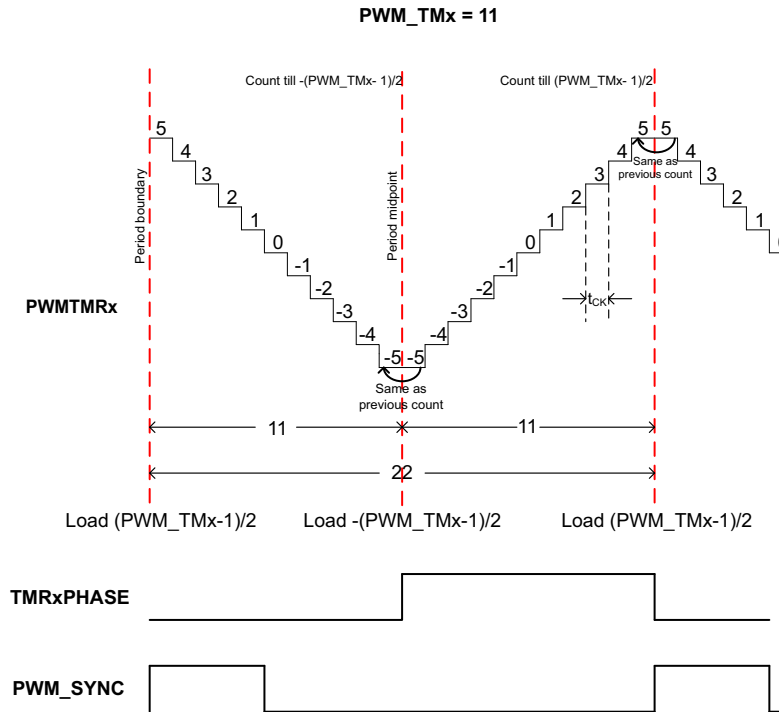


**Figure 18-2:** Operation of Timer for Odd Value of PWM\_TM

When the timer register value is even, for example 12, then that timer loads +5 at the beginning of the period, counts from +5 to -6 in the first half, reloads -5 at the midpoint and counts up from -5 to +6 in the second half. The reload values at the period and mid-period boundaries are different from the previous count. It totally counts  $2 \times 12 = 24$  counts in the entire period. This is shown in the figure below. Note that both half-periods have a count of 12 each.

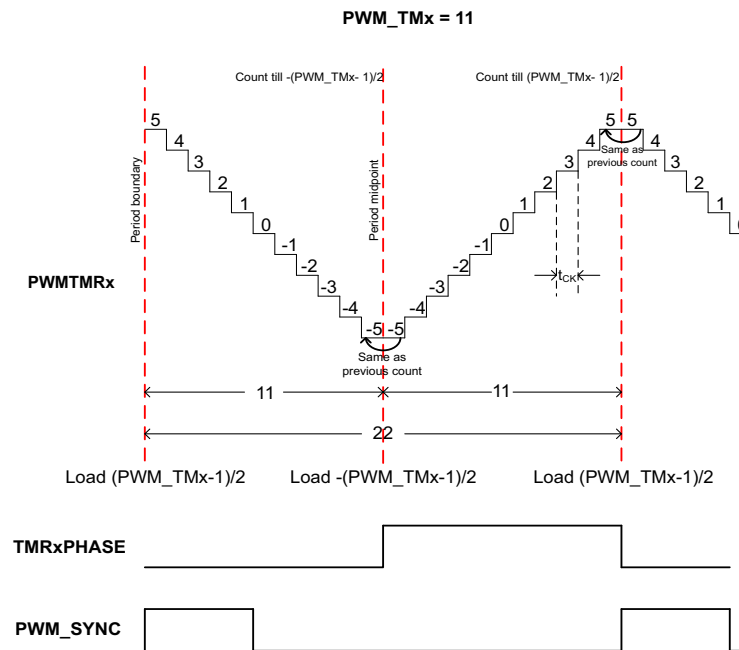


**Figure 18-3:** Operation of Timer for Even Value of PWM\_TM

Note that in the operation discussed in this section, double buffering of all channel registers and the timer registers take place at the period boundary of the respective timers.

## Phase Offset Control

The PWM timers (PWMTMR1 through PWMTMR4) can operate with a programmable phase lag with respect to the main timer, PWMTMR0. Using DLY x delay counter in conjunction with PWMTMR0 and setting the  $PWM\_CTL.DLYAEN$  bit implements this feature for a given channel.

If this feature is enabled for channel A (and channel A is using PWMTMR1 for generating duty cycle), when PWMTMR0 reaches its period boundary, it triggers, DLY A, which counts out SCLK cycles equal in number to the value programmed in the  $PWM\_DLYA$  register. At the end of this count, DLY x sends out a trigger to PWMTMR1. Thus PWMTMR1 receives a synchronization pulse in every period of PWMTMR0 at a point delayed from its period boundary by the value in  $PWM\_DLYA$  register. For more information on how channels can reference different timers for their outputs, see "Channel Timing Control Unit"

**NOTE:** The following conditions must be satisfied when this feature is used on PWMTMRy for channel Y with respect to PWMTMRx.

- The  $PWM\_DLYA$  register must be programmed to a value less than  $2 \times TMy$
- $PWM\_TM0 = N \times PWM\_TMy$  where N is an integer.

The function of PWMTMR<sub>y</sub> (PWMTMR1 in the example above) differs in cases where  $PWM\_TM0 = PWM\_TM1$  from cases where  $PWM\_TM0 = N \times PWM\_TM1$ . Both cases are examined below.

#### **Case 1: $PWM\_TM0 = PWM\_TM_y$**

In this case every period of PWMTMR<sub>y</sub> starts its period again, on reception of the synchronization pulse from DLY-x. If the trigger from DLY-x is late, it holds its count till the trigger comes. If the trigger is a bit early, it reloads without regard to whether it has completed its current period. Thus PWMTMR<sub>y</sub> is re-synched with PWMTMR0 with a phase lag as programmed in PWM\_DLYA register, in every one of its periods.

Note that in this case the expiration of the DLY x counter is the period boundary of PWMTMR<sub>y</sub>. Therefore, this is the point of update of all the double buffered registers related to the given channel (except the delay registers which are double buffered at the period boundary of PWMTMR0).

*Phase Offset Control Using DELAY* shows an example where:

- PWM\_TM0, PWM\_TM1 and PWM\_TM2 are programmed with the same value.
- PWM\_DLYA and PWM\_DLYB are programmed values DELAY1 and DELAY2 respectively, such that  $DELAY2 > DELAY1$ .
- Channel A's outputs are referenced to TMR1 and Channel B's outputs to PWMTMR2.

The delay registers are double buffered and the new value of delay is reloaded at the period boundary of PWMTMR0. The two options described below exist if the new value is different from the older one. The behavior of PWMTMR<sub>y</sub> in both these cases is discussed, in conjunction with *Impact of New DELAY Value on Timer Count for Equal TM*. It is assumed that channel B references its outputs to PWMTMR0 and channel A references its outputs to PWMTMR1.

1. The new delay value is higher than the previous value. Therefore, the corresponding PWMTMR<sub>y</sub> is allowed more than one period's time between consecutive triggers from the DLY-x counter. In this case, after reaching its period boundary, PWMTMR<sub>y</sub> holds its count at the period boundary and waits for the trigger from DLY-x. This is shown in CASE A in *Impact of New DELAY Value on Timer Count for Equal TM*.
2. The next delay value programmed is smaller than the previous value. In this case, the corresponding PWMTMR<sub>y</sub> is allowed only less than one period's time between consecutive triggers from the DLY-x counter. Though the trigger comes earlier in this case, before PWMTMR<sub>y</sub> has counted out one full period, it reloads and starts its period again. This is shown in CASE B in *Impact of New DELAY Value on Timer Count for Equal TM*.

Therefore, PWMTMR1 waits and obeys a synchronization pulse from DLY A in every one of its periods.



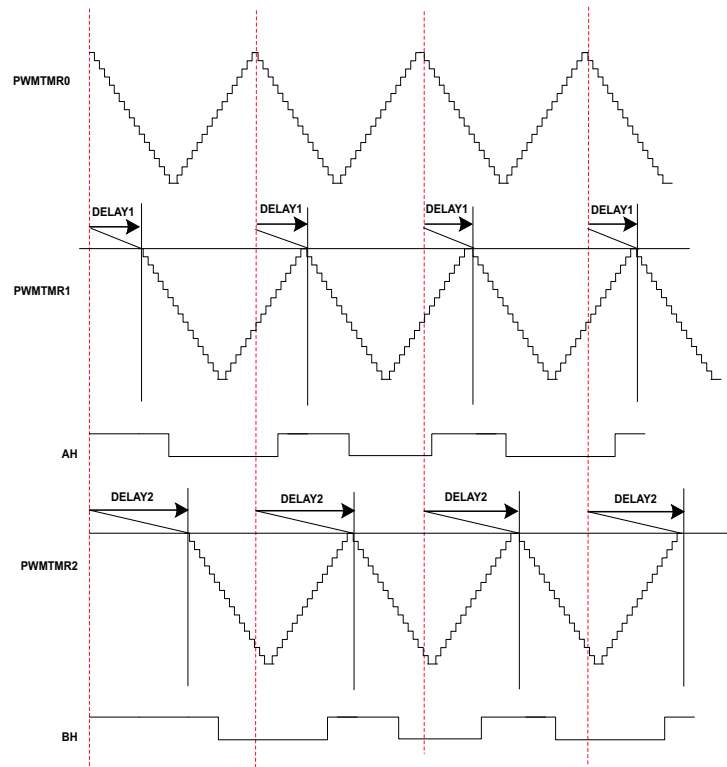


**Figure 18-4:** Phase Offset Control Using DELAY

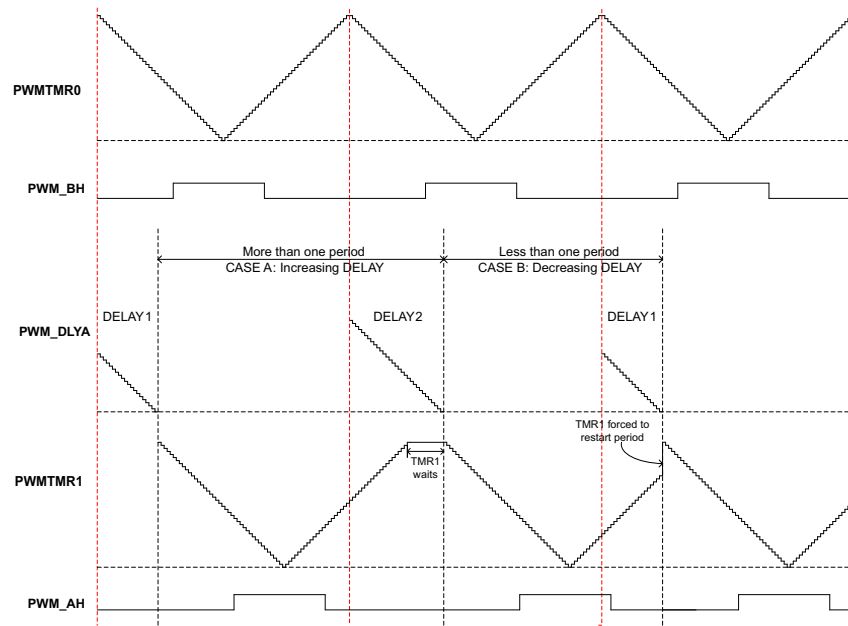


**Figure 18-5:** Impact of New DELAY Value on Timer Count for Equal TM

### Case 2: $PWM\_TM0 = N \times PWM\_TM_y$

In this case, within a single period of PWM\_TMR0 a program can fit multiple periods (N) of PWM\_TMRy. Additionally, the DLY-x counter is triggered only once every N periods of PWM\_TMRy.

The operation is as follows: Every Nth period of PWM\_TMRy, it expects a synchronization pulse from DLY-x counter. When it counts out that period if the trigger hasn't yet arrived yet, it waits at the end of the period for the trigger, and starts counting down once it arrives. If the trigger comes earlier than that, it restarts immediately without waiting to complete its period count.

In the intervening periods, PWM\_TMRy operates independently. As the period ends, it reloads and starts the next period without intervention from the delay counter.

*Impact of DELAY Value Change for the Multiple TM* gives an example with  $N = 2$ . Note that PWM\_TMRy syncs up with PWM\_TMR0 every 2nd period, and is free running across every odd period boundary.

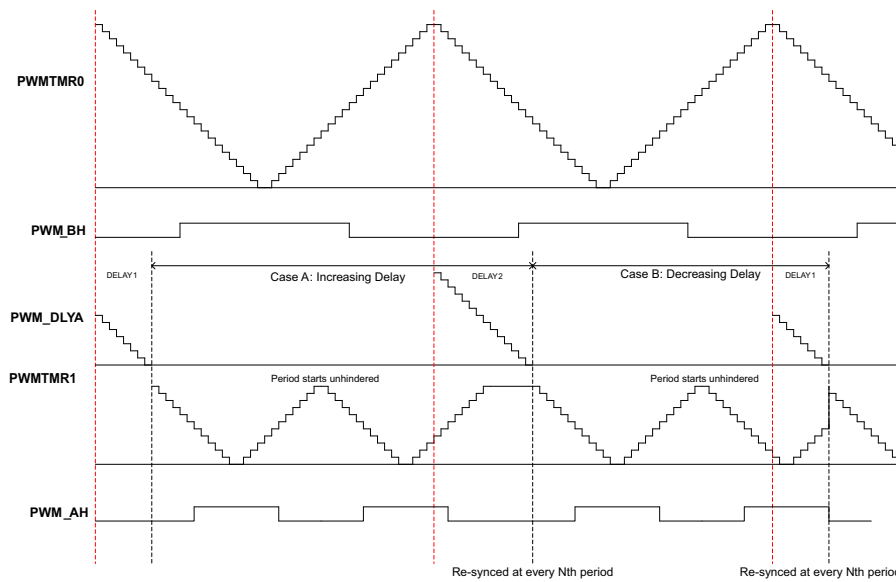


**Figure 18-6:** Impact of DELAY Value Change for the Multiple TM

## Channel Timing Control Unit

The channel timing control unit is the core of the PWM. There are four separate channels, each channel controlling a pair of output signals – the high side output and the low side output.

## Channel Control

The `PWM_CHANCFG` register controls the static configuration of all the channels and is to be initialized once before the beginning of the operation.

**NOTE:** The `PWM_CHANCFG` register is not double buffered and the contents of it must not be changed once the PWM is enabled.

Each channel works off a reference timer base. The time base can be either the main timer `PWMTMR0` or the appropriate `PWMTMRx` as given below. This can be configured with the `PWM_CHANCFG.REFTMRA` bit field.

- Channel A can work off `PWMTMR0` or `PWMTMR1`
- Channel B can work off `PWMTMR0` or `PWMTMR2`
- Channel C can work off `PWMTMR0` or `PWMTMR3`
- Channel D can work off `PWMTMR0` or `PWMTMR4`

The `PWM_xCTL` registers contain bits which control the dynamic pulse behavior of the channel outputs and this register is double buffered. This register has bits which control enable/disable and pulse positioning of the outputs. The impact of these is explained in the following sections.

## Pulse Positioning and Duty Cycle Registers

The `PWM_ACTL.PULSEMODEHI[1:0]` and `PWM_ACTL.PULSEMODELO[1:0]` fields define the region within the timer period where the output pulses should be positioned. When the `PWM_CHANCFG.MODELSC` bit is 0, the `PWM_ACTL.PULSEMODEHI` field specifies the pulse positioning for both the high-side and low-side outputs of the channel. When the bit is 1, `PWM_ACTL.PULSEMODELO` defines the pulse positioning for the low side channel output, while `PWM_ACTL.PULSEMODEHI` continues to do serve this purpose for the high side channel output.

Two Duty-Cycle registers are provided for each channel output: `PWM_AH0` and `PWM_AH1` for the high side output, and `PWM_AL0` and `PWM_AL1` for the low side output. These registers determine the width of the output pulses. When the `PWM_CHANCFG.MODELSC` bit is 0, the high side Duty-Cycle registers are used for the low side output pulse width determination as well. The duty cycle range that can be programmed into these registers is between  $-PWM_{TMx}/2$  and  $+PWM_{TMx}/2$  when dead-time is not considered.

When dead-time is considered, for `PULSEMODE 00` and `01`, the programmed duty is modified for use by the PWM in such a way that the range is limited between the values  $[-PWM_{TMx}/2 + PWM_{DT}]$  to  $[+PWM_{TMx}/2 + PWM_{DT}]$  considering the high-side output. Dead-time is explained in detail in a later section. For `PULSEMODEs 10` and `11`, the high-side Duty-Cycle registers are limited between  $[PWM_{TMx}/2 + PWM_{DT}]$  and  $[-PWM_{TMx}/2 - PWM_{DT}]$ .

**NOTE:** Values programmed into these registers that fall outside these limits result in over/under modulation.

## Duty Cycle and Pulse Positioning Control

The `PWM_ACTL.PULSEMODEHI` and `PWM_ACTL.PULSEMODELO` fields control how the Duty cycle registers modify the waveform of the high and low side outputs. (The `PWM_ACTL.PULSEMODEHI` and `PWM_ACTL.PULSEMODELO` fields are referred to as *PULSEMODE* in the subsequent discussion.)

- `PULSEMODE = 00` – Allows symmetrical pulse waveform around the center of the PWM period. In this mode, only one of the Duty Cycle registers is used for an output—for example, for the AH output, only the `PWM_AH0` register is used. Note that in this mode, the values in the duty cycle registers are scaled such that a value of 0 produces 50% duty.
- `PULSEMODE = 01` – Allows asymmetrical pulse waveform around the center of the PWM period. In this mode both the Duty Cycle registers are used. For example for the AH output, `PWM_AH0` and `PWM_AH1` registers are used. In this mode, if `PWM_AH1` is programmed with the same value as `PWM_AH0`, the obtained output is identical to that in the `PULSEMODE = 00` case.
- `PULSEMODE = 10` or `11` – Allow pulse waveforms to be produced either on the first half or the second half of the PWM period respectively, and both `PWM_AH0` and `PWM_AH1` registers are used.

**NOTE:** In `PULSEMODE = 10` the condition `PWM_AH0 > PWM_AH1`. If the low side is working off the low side Duty-Cycle registers, the condition `PWM_AL0 > PWM_AL1` should be strictly adhered to.

**NOTE:** In `PULSEMODE = 11` the condition `PWM_AH0 < PWM_AH1`. If the low side is working off the low side Duty-Cycle registers, the condition `PWM_AL0 < PWM_AL1` should be strictly adhered to.

The following figure shows the pulse positioning modes as described above for PWM\_AH. In the figure, DUTY0 is assumed to be the value in PWM\_AL0 register and DUTY1, the value in PWM\_AH1 register. The step signal, count, indicates the output of the timer that Channel A is working off. Also, in the example, the signal is assumed to be configured as active high, and dead-time is assumed to be zero.

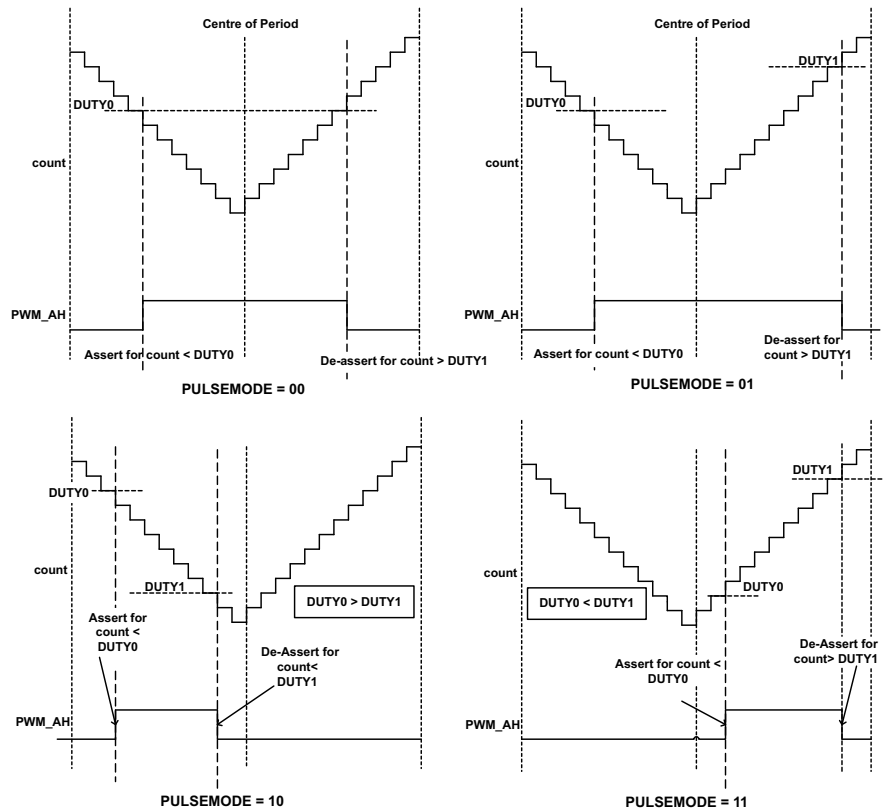


**Figure 18-7:** Pulse Positioning Modes

## Channel Low Side Output Dependent Operation Mode and Dead-Time

The low-side output waveform can be programmed to be dependent on the high side output waveform or be totally independent. This is controlled using the PWM\_CHANCFG.MODELSC bit.

For example, the channel A produces the high side output PWM\_AH and the low side output PWM\_AL. When PWM\_CHANCFG.MODELSC = 0, the low-side output is also generated using the high-side duty-cycle registers for pulse width, PWM\_ACTL.PULSEMODEHI bits for pulse positioning and the PWM\_CHANCFG.POLAH bit for polarity. If PWM\_DT is programmed to 0, the low-side output is an inverted version of the high-side output.

When PWM\_DT is programmed with a non-zero value, both the high-side and low-side outputs are shrunk symmetrically about the points of transition in the zero dead-time case by an amount PWM\_DT.

The high and low-side outputs for the case with zero and non-zero dead-time for PWM\_ACTL.PULSEMODEHI = 00 and 01 are shown in the following figure. In the figure, DUTY0 is the value programmed into the

PWM\_AH0 register and DUTY1 is the value programmed into the PWM\_AH1 register. PWM\_CHANCFG.POLAH is 1 indicating that both signals are active high. PWM\_DT holds the value DT.

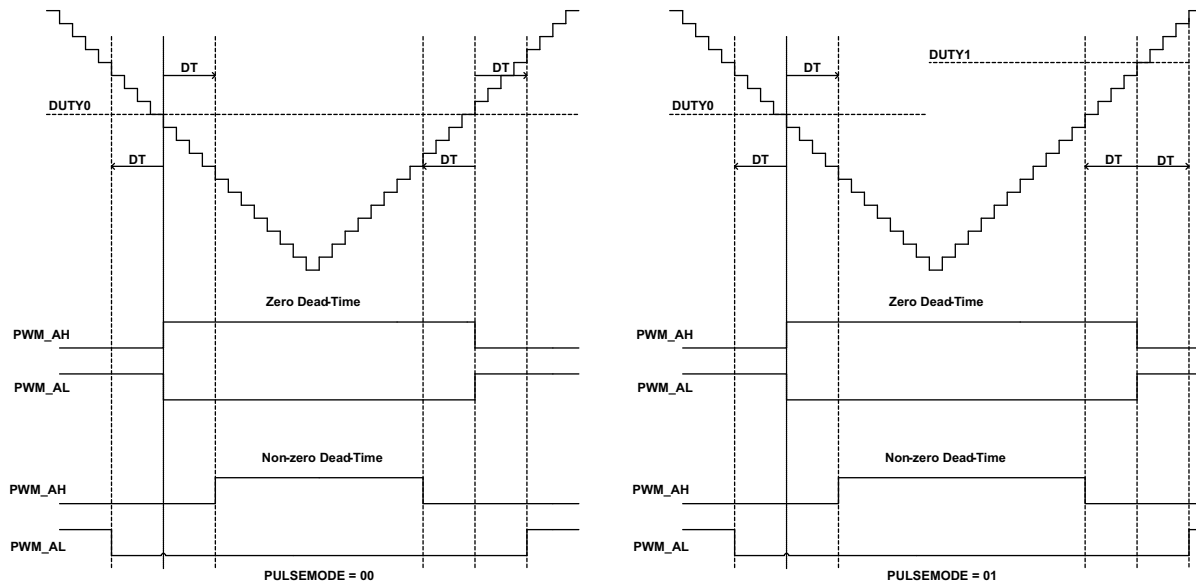


**Figure 18-8:** Channel Outputs in Dependent Mode for PULSEMODE = 00, 01

The high and low-side outputs for the case with zero and non-zero dead-time for PWM\_ACTL.PULSEMODEHI = 10 and 11 are shown in the following figure. In the figure, DUTY0 is the value programmed into PWM\_AH0 register and DUTY1 is the value programmed into the PWM\_AH1 register. PWM\_CHANCFG.POLAH is 1 indicating that both signals are active high. PWM\_DT holds the value DT.

**NOTE:** Bringing dead-time into the picture, the guidelines for programming the Duty-Cycle registers in PULSEMODEs 10 and 11 given in “PWM Duty Cycle and Pulse Positioning Control” are modified as follows:

PULSEMODE 10:  $\text{PWM\_xH0} - \text{DT} > \text{PWM\_xH1} + \text{DT}$

PULSEMODE 11:  $\text{PWM\_xH0} + \text{DT} < \text{PWM\_xH1} - \text{DT}$

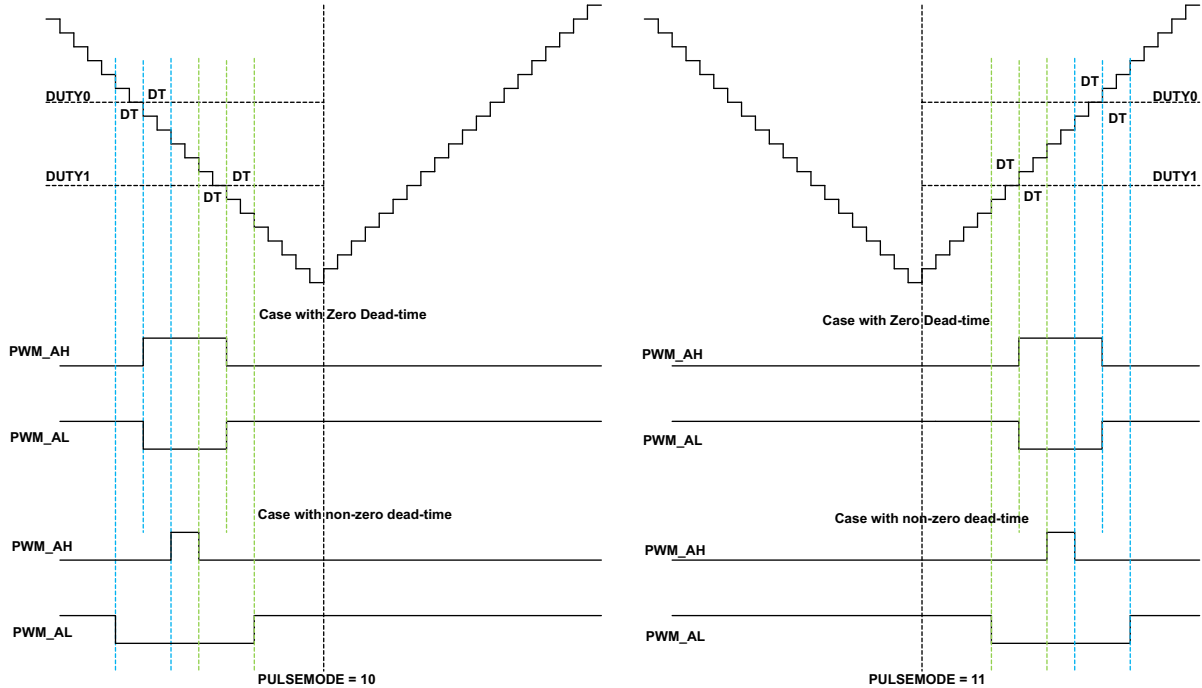


**Figure 18-9:** Channel Outputs in Dependent Mode for PULSEMODE = 10, 11

## Channel High Side and Low Side Outputs, Independent Operation Mode

Independent control of the channel outputs, PWM\_AH0 and PWM\_AL0 is possible by setting the PWM\_CHANCFG.MODELSA bit to 1. In this case, PWM\_AH is generated using PWM\_AH0, PWM\_AH1 for pulse width, PWM\_ACTL.PULSEMODEHI for pulse position, and PWM\_CHANCFG.POLAH for polarity. PWM\_AL is generated using PWM\_AL0, PWM\_AL1 for pulse width, PWM\_ACTL.PULSEMODELO for pulse position and PWM\_CHANCFG.POLAL for polarity.

**NOTE:** In the independent mode, the dead-time insertion is not applicable and the dead-time is forced to zero by the hardware.

The following figure shows an example of the independent mode of operation where PWM\_AH and PWM\_AL work off different register bits.

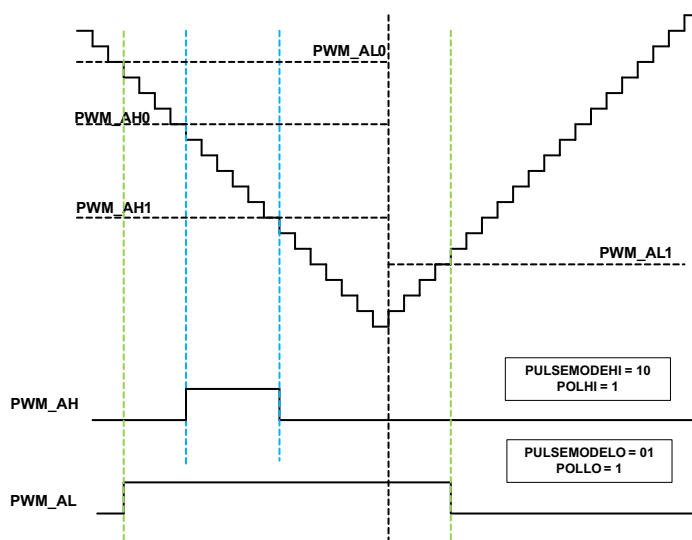


**Figure 18-10:** PWM\_AH and PWM\_AL in Independent Operation Mode

Note that PWM\_AH and PWM\_AL can be positioned in the timer period with a given phase difference between them. This is achieved by having PULSEMODEHI and PULSEMODELO bits programmed to different values. This is shown in the next figure.

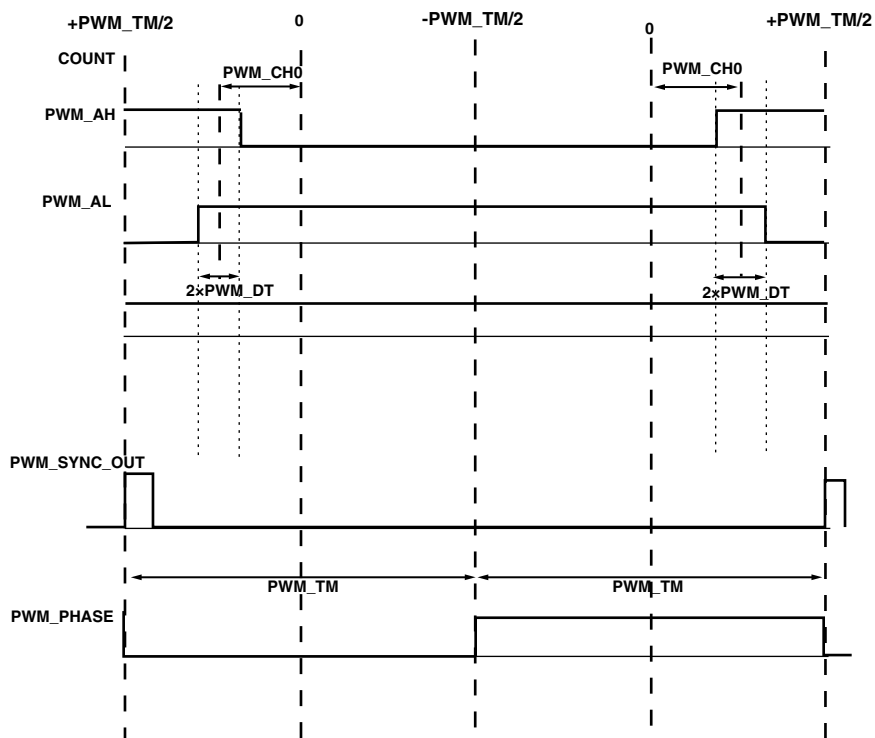


**Figure 18-11:** Channel Outputs Controlled Independently



## Switched Reluctance Motors Application

In the typical power converter configuration for switched or variable reluctance motors, the motor winding is connected between the two power switches of a given inverter leg. Therefore, to allow for a complete circuit in the motor winding, it is necessary to turn on both switches at the same time.

Switched reluctance motors are used in the following configurations: Hard Chop, Alternate Chop, Soft Chop—Bottom On, and Soft Chop—Top On.

The following figure shows the four SR mode types as active high PWM output signals.

Hard Chop mode contains independently programmed rising edges of a channel's high and low signals in the same PWM half cycle and both contain independently programmed falling edges in the next PWM half cycle. The `PWM_CHANCFG.POLAH` and `PWM_CHANCFG.POLAL` are programmed to same values.

Alternate Chop mode is similar to normal PWM operation but the PWM channel high and low signal edges are always opposite and are independently programmed. The `PWM_CHANCFG.POLAH` and `PWM_CHANCFG.POLAL` are programmed to opposite values. The Low Side Invert is the only difference between Hard Chop mode and Alternate Chop mode.

Soft Chop—Bottom On uses a 100% duty on the low side of the channel and Soft Chop—Top On uses a 100% duty on the high side of the channel. Similar to Hard Chop mode the `PWM_AH0` duty register is used for the high channel and `PWM_AL0` duty register is used for the low channel.

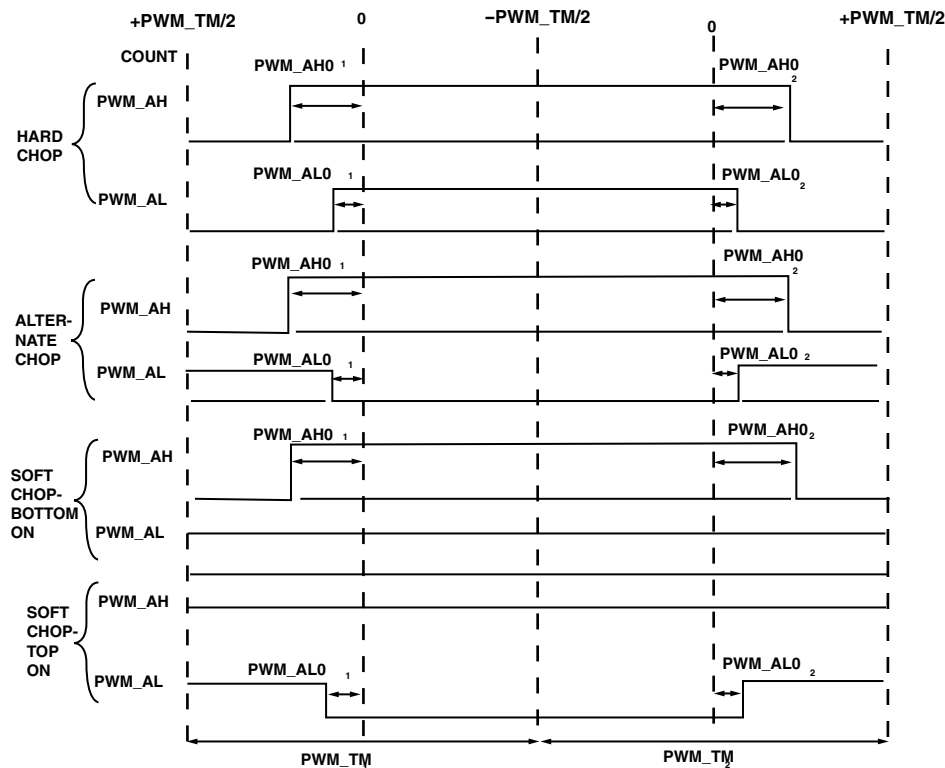


**Figure 18-12:** Four SR Mode Types, Active High PWM Output Signals

## Switching Dead Time (PWM\_DT) Register

The second important parameter that must be set up in the initial configuration of the PWM Controller is the switching dead time. This is a short delay introduced between turning off one PWM signal (for example, AH) and turning on the complementary signal (for example, AL). This short time delay permits the power switch being turned off (AH in this case) to completely recover its blocking capability before the complementary switch is turned on. This time delay prevents a potentially destructive short-circuit condition from developing across the dc link capacitor of a typical voltage source inverter.

The 10-bit, read/write PWM\_DT register controls the dead time. This register controls the dead time inserted into the three pairs of PWM output signals. Dead time ( $T_d$ ) is related to the value in the PWM\_DT register by:

$$T_d = \text{PWM\_DT} \times 2 \times t_{\text{SCLK}}$$

Therefore, a PWM\_DT value of 0x00A introduces a 200 ns delay (for a SCLK of 100 MHz) between turning off any PWM signal (for example, AH) and then turning on its complementary signal (for example, AL). The length of the dead time can therefore be programmed in increments of  $2 \times t_{\text{SCLK}}$  (or 20 ns for an SCLK of 100 MHz). The PWM\_DT register is a 10-bit register whose maximum value of 0x3FF (1023 decimal) corresponds to a maximum programmed dead time of:

$$T_{d(\text{max})} = 1023 \times 2 \times t_{\text{SCLK}} = 1023 \times 2 \times 10 \times 10^{-9} = 20.5 \mu\text{s}$$

for an  $f_{\text{SCLK}}$  rate of 100 MHz. The dead time can be programmed to be zero by writing 0 to the PWM\_DT register.

## Duty Cycle with Dead-Time Control: Calculations for PULSEMODE 00

The duty cycle registers are scaled such that a value of 0 represents a 50% PWM duty, cycle. The switching signals produced are also adjusted to incorporate the programmed dead time value in the PWM\_DT register. The unit in this case produces active low signals so that a low level corresponds to a command to turn on the associated power device.

A typical pair of PWM outputs, PWM\_AH and PWM\_AL, is shown in the following figure. The time values in the figure indicate the integer value in the associated register and can be converted to time by multiplying by the fundamental time increment,  $t_{\text{CK}}$ . In the example channel A is working off of PWMTMR0.

Because PULSEMODE is set to 00, the switching patterns are perfectly symmetrical about the mid-point of the switching period. The dead time is incorporated by moving the switching instants of both PWM signals away from the instant set by the PWM\_AH0 register. Both switching edges are moved by an equal amount ( $\text{PWMDT} \times t_{\text{CK}}$ ) to preserve the symmetrical output patterns. Also shown is the PWM\_SYNC\_OUT pulse whose rising edge denotes the beginning of the switching period, and the PWM\_STAT.TMR0PHASE bit.

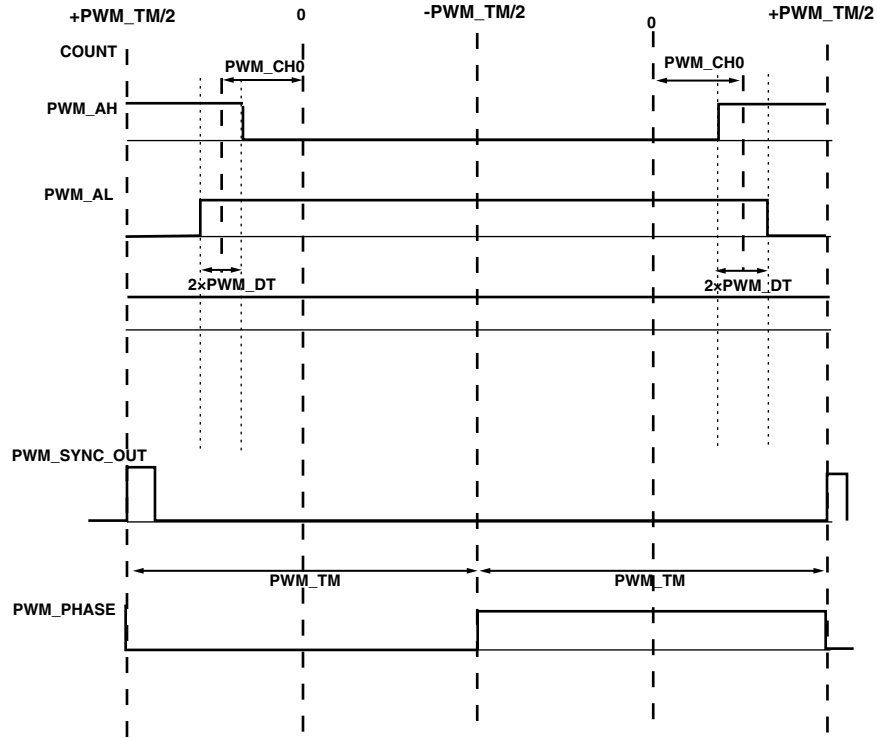


**Figure 18-13:** Dead-Time Between Outputs in Dependent Mode

The resulting on-times (active low) of the PWM signals over the full PWM period (two half-periods) produced by the PWM timing unit and illustrated in the figure above, may be written as shown in the following equation.

$$T_{AH} = (PWM\_TM0 + 2 \times (PWM\_AH0 - PWM\_DT)) \times t_{CK};$$

$$\text{Range of } T_{AH} \text{ is } [0:2 \times PWM\_TM0 \times t_{CK}]$$

$$T_{AL} = (PWM\_TM0 - 2 \times (PWM\_AH0 + PWM\_DT)) \times t_{CK};$$

$$\text{Range of } T_{AL} \text{ is } [0:2 \times PWM\_TM0 \times t_{CK}]$$

The corresponding duty cycles are shown in the following equations.

$$d_{AH} = \frac{T_{AH}}{T_s} = \frac{1}{2} + \frac{PWM\_AH0 - PWM\_DT}{PWM\_TM}$$

$$d_{AL} = \frac{T_{AL}}{T_s} = \frac{1}{2} - \frac{PWM\_AH0 + PWM\_DT}{PWM\_TM0}$$

The negative values of  $T_{AH}$  and  $T_{AL}$  are not permitted and the minimum permissible value is zero, corresponding to a 0% duty cycle. In a similar fashion, the maximum value is  $T_s$ , the PWM switching period, corresponding to a 100% duty cycle. Calculation of duty for other PULSEMODOs can be similarly carried out.

## Special Consideration for PWM Operation in Over-Modulation

The PWM Timing Unit is capable of producing PWM signals with variable duty cycle values at the PWM output pins. In PULSEMODEs 00 and 01, at the extremities of the modulation process, we have 0% and 100%. IN PULSEMODEs 01 and 10, at the extremities the modulation process, we have 0% and 50%. The modulation is called FULL OFF when the lower extremity of modulation is set for any PWM timer period for the corresponding channel. The modulation is called FULL ON when the higher extremity of modulation is set for any PWM timer period for the corresponding channel. In between, for other duty cycle values, the operation is termed NORMAL MODULATION.

- Full On Modulation. In PULSEMODEs 00 and 01, a PWM channel is said to be in FULL ON modulation if the high-side output of that channel is asserted for the whole duration of the period of the PWM timer that channel is referencing. Condition for FULL ON MODULATION:  $PWM\_xH0 - DT > PWM\_TMy/2$  for PULSEMODE 00 and the additional condition that  $PWM\_xH1 - DT > PWM\_TMy/2$  as well.

In PULSEMODE 10, a PWM channel is said to be in FULL ON modulation if the high-side output of that channel is asserted for the whole duration of the first half period of the PWM timer that the channel is referencing. Condition for FULL ON MODULATION:  $PWM\_xH0 - DT > PWM\_TMy/2$  and  $PWM\_xH1 + DT < -PWM\_TMy/2$ .

In PULSEMODE 11, a PWM channel is said to be in FULL ON modulation if the high-side output of that channel is asserted for the whole duration of the second half period of the PWM timer that the channel is referencing. Condition for FULL ON MODULATION:  $PWM\_xH0 + DT < -PWM\_TMy/2$  and  $PWM\_xH1 - DT > PWM\_TMy/2$ .

- Full Off Modulation. In PULSEMODEs 00 and 01, a PWM channel is said to be in FULL OFF modulation if the high-side output of that channel is de-asserted for the whole duration of the period of the PWM timer that channel is referencing. Condition for FULL OFF MODULATION:  $PWM\_xH0 - DT < -PWM\_TMy/2$  for PULSEMODE 00 and the additional condition that  $PWM\_xH1 - DT < -PWM\_TMy/2$  as well.

In PULSEMODE 10, a PWM channel is said to be in FULL OFF modulation if the high-side output of that channel is de-asserted for the whole duration of the first half period of the PWM timer that the channel is referencing (In the second half-period it is anyway de-asserted). Condition for FULL OFF MODULATION:  $PWM\_xH0 - DT < -PWM\_TMy/2$  and  $PWM\_xH1 + DT < PWM\_xH0 - DT$ .

In PULSEMODE 11, a PWM channel is said to be in FULL OFF modulation if the high-side output of that channel is de-asserted for the whole duration of the second half period of the PWM timer that the channel is referencing (In the first half of the period it is anyway de-asserted). Condition for FULL OFF MODULATION:  $PWM\_xH0 + DT > PWM\_TMy/2$  and  $PWM\_xH1 - DT > PWM\_xH0 + DT$ .

- Normal Modulation. All other cases of modulation fall under this category.

There are certain situations, on transition either into or out of either FULL ON or FULL OFF modulation, where it is necessary to insert additional emergency dead time delays to prevent potential shoot through conditions in the inverter. Disable/Enable usage (related to DISHI and DISLO bits in PWM\_CTL register) also can potentially cause outputs to violate shoot through condition criteria. Another case is when the

phase delay of a PWM timer is varied by large values. These transitions are detected automatically and if appropriate for safety, an emergency dead-time is inserted to prevent shoot through conditions.

The insertion of the additional emergency dead time into one of the PWM signals of a given pair during these transitions is only needed if otherwise both PWM signals would be required not to toggle within a dead time of each other. The additional emergency dead time delay is inserted into the PWM signal that is toggling into the ON state. In effect the turn ON of this signal is delayed by an amount ( $2 \times \text{PWMDT} \times t_{\text{CK}}$ ) from the rising edge of the opposite output. After this delay, the PWM signal is allowed to turn ON provided the desired output is still scheduled to be in the ON state after the emergency dead time delay.

The following figure illustrates two example of such a transition. Here POLHI is kept at 1. PWM\_AH has been in FULL ON modulation for sometime, and during the current period, its PULSEMODE is changed to 10 keeping the FULL ON condition. At the half-period boundary it is forced to transition to a de-asserted state due to this being PULSEMODE 10. It can be seen that an emergency dead-time is inserted on the low-side output.

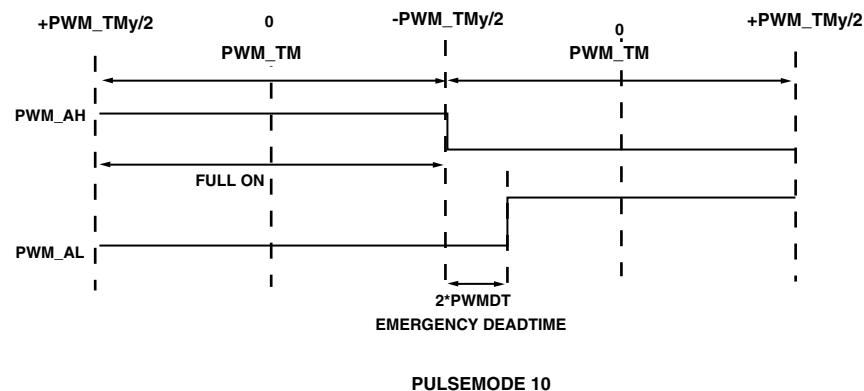


Figure 18-14: Over Modulation Transition Scenarios

## Output Disable and Cross-Over Functions

Each PWM\_ACTL channel control register contains separate enable bits for the high and low side signals. For example, the PWM\_ACTL.DISHI and PWM\_ACTL.DISLO bits in the channel A control register control the enable/disable of the AH and AL outputs respectively. If the disable bit is set ( $= 1$ ), then the corresponding PWM output is disabled, irrespective of the value of the corresponding duty cycle register. This PWM output signal remains in the OFF state as long as the corresponding enable/disable bit is set.

The PWM\_ACTL.XOVR bit, or the cross-over bit allows programs to send the low-side output through the high-side output pin and the high-side output through the low side output pin.

For example, PWM\_AH0 is programmed to zero with PWM\_CHANCFG.MODELSA = 0. With PWM\_ACTL.DISLO = 1 and PWM\_ACTL.XOVR = 1 the low-side is being toggled even though the output is disabled, but nothing is toggling on the high-side pins. What is happening is that the high-side output is now coming out through the low-side (and vice versa). What is toggling on the low-side pin is the 50% duty high side output pulse (less the dead-time if any).

The following figure shows this example. In case 1 PWM\_ACTL.XOVR = 0 and in case 2 PWM\_ACTL.XOVR = 1.

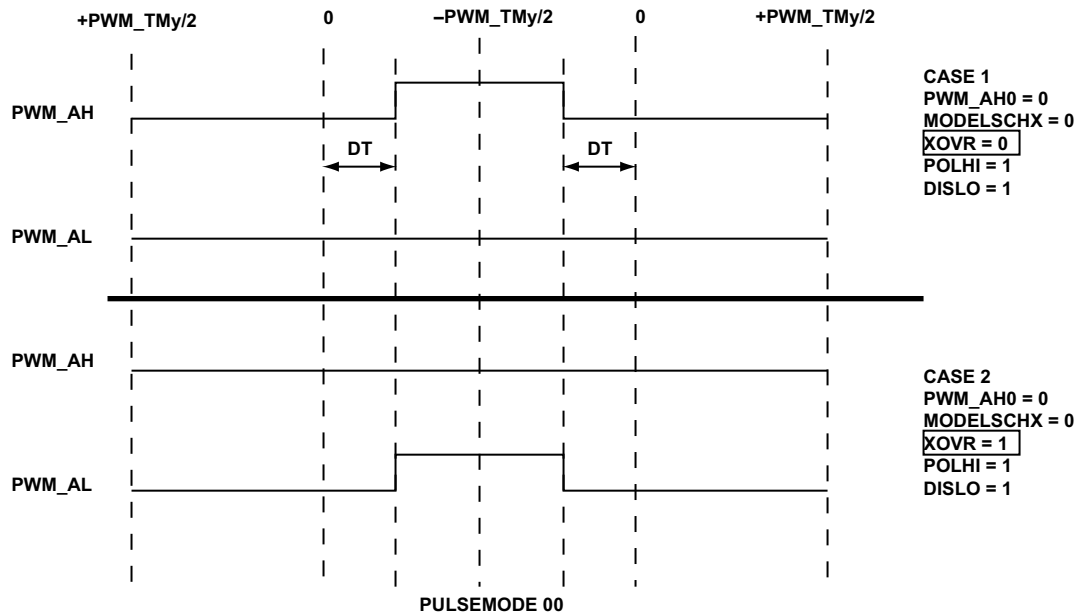


**Figure 18-15:** XOVR and DISHI/DISLO Functionality

## Brushless DC Motor (Electronically Commutated Motor) Control

In the control of an electronically commutated motor (ECM), only two inverter legs are switched at any time. Often, the high-side device in one leg must be switched on at the same time as the low-side driver in a second leg. Therefore, by programming identical duty cycles values for two PWM channels (for example,  $PWM\_CHA = PWM\_CHB$ ) and setting the  $PWM\_BCTL.XOVR$  bit to crossover the BH/BL pair if PWM signals, it is possible to turn on the high-side switch of phase A and the low-side switch of phase B at the same time.

To control ECM, normally the third inverter leg (phase C in this example) is disabled for a number of PWM cycles. To implement this function, both the  $PWM\_CH$  and  $PWM\_CL$  outputs are disabled by setting the  $PWM\_CTL.DISHI$  and  $PWM\_CTL.DISLO$  bits.

In normal ECM operation, each inverter leg is disabled for certain time periods so that the PWM channel registers change based on the position of the rotor shaft (motor commutation).

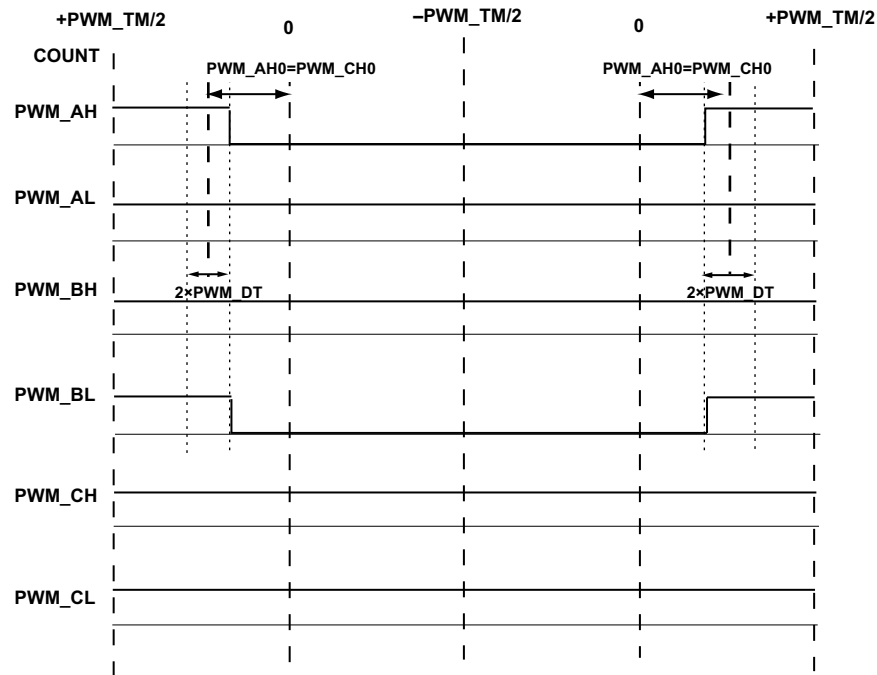


**Figure 18-16:** ECM Control

For the situation illustrated in the figure, an appropriate value for the `PWM_SEG` register is 0x00A7. In normal ECM operation, each inverter leg is disabled for certain lengths of time, such that the `PWM_SEG` register is changed, based upon the position of the rotor shaft (motor commutation).

## Gate Drive Unit

The Gate Drive Unit of the PWM adds features that simplify the design of isolated gate drive circuits for PWM inverters. If a transformer coupled power device gate drive amplifier is used then the active PWM signal must be chopped at a high frequency. The `PWM_CHOPCFG` register allows the programming of this high frequency chopping mode. The chopped active PWM signals may be required for the high-side drivers only, for the low-side drivers only or for both the high-side and low-side switches. Therefore, independent control of this mode for both high and low-side switches is included with two separate control bits in the `PWM_CHANCFG` register.

Typical PWM output signals with high-frequency chopping enabled on both high-side and low-side signals are shown in the figure below. Chopping of the PWM outputs is enabled by setting bits in `PWM_CHANCFG` register. The high frequency chopping frequency is controlled by the 8-bit `PWM_CHOPCFG.VALUE` value in the `PWM_CHOPCFG` register. The period of this high frequency carrier is then given by,

$$T_{chop} = [4 \times (CHOPDIV + 1)] \times t_{CK}$$

and the chopping frequency is therefore an integral subdivision of the peripheral clock frequency:

$$f_{\text{chop}} = f_{\text{CK}} / [4 \times (\text{CHOPDIV} + 1)]$$

The PWM\_CHOPCFG.VALUE value may range from 0 to 255, corresponding to a programmable chopping frequency rate from 122 kHz to 31.25 MHz for a 125 MHz,  $f_{\text{CK}}$  rate. The gate drive features must be programmed before operation of the PWM controller and are not changed during normal operation of the PWM controller. Following a reset, all bits of the PWM\_CHANCFG register are cleared so that high frequency chopping is disabled, by default.

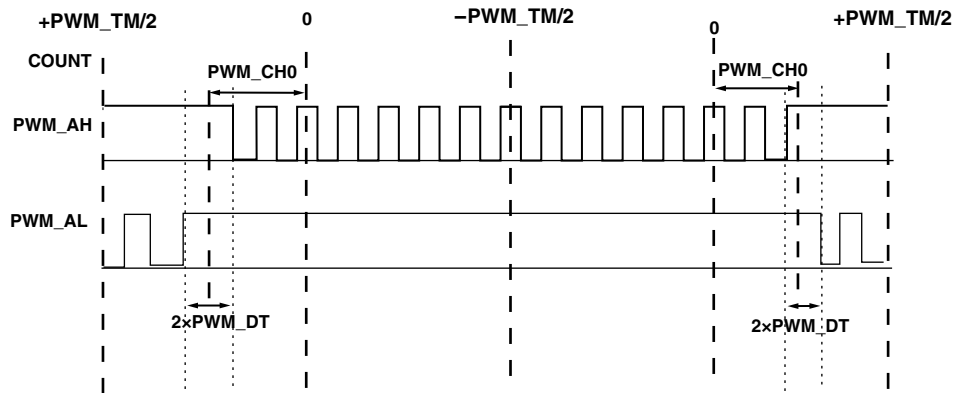


**Figure 18-17:** Hi-Side and Lo-Side Outputs With Gate Chop Enabled

## Output Control Feature Precedence

It is important to understand the order in which output control features are applied to the PWM signal. The following hierarchy indicates the order (from most important to least important) in which signal features are applied to the PWM output signal.

1. Channel duty generation
2. Channel Crossover
3. High-side/Low-side disable
4. Emergency dead time insertion
5. Gate drive chopping
6. Polarity

## Sync Operation

The PWM\_SYNC signal can be internally generated as a function of PWM\_TMO.VALUE and PWM\_SYNC\_WID.VALUE or can be input externally. Multiple PWM configurations can be established with each PWM operating with its own independent PWM\_SYNC or from its own or shared external PWM\_SYNC signal. The external PWM\_SYNC can be synchronous to the internal clock as in the case of a primary PWM



generating an internal PWM\_SYNC signal which drives the secondary PWM\_SYNC\_IN pin. The external PWM\_SYNC can also be asynchronous to the internal clock as is typically the case of an off-chip PWM\_SYNC signal used to drive each PWM's PWM\_SYNC\_IN pin.

## Internal PWM SYNC Generation

The PWM controller produces an output PWM synchronization pulse at a rate equal to period of a selected PWM timer. Programming the PWM\_CTL.INTSYNCREF field controls this selection.

Further, if the other timers are running with a non-zero DELAY offset in relation to PWMTMR0 and the PWM\_SYNC pulse is referenced to any of these timers, then the PWM\_SYNC pulses are generated at their respective period boundaries which has the lag-lead offset compared to PWMTMR0.

This pulse is available for external use at the PWM\_SYNC\_OUT pin. The width of the PWM\_SYNC pulse is programmable by the 10-bit read/write PWM\_SYNC\_WID register. The width of the PWM\_SYNC pulse,  $t_{PWM\_SYNC}$ , is given by the following equation.

$$t_{PWM\_SYNC} = t_{CLK} \times (PWMSYNCWT + 1)$$

The width of the pulse is programmable from  $t_{CLK}$  to  $1024t_{CLK}$  (corresponding to 8 ns to 8.19  $\mu$ s for a  $f_{CLK}$  rate of 125 MHz). Following a reset, the PWM\_SYNC\_WID register contains 0x3FF (1023 decimal) so that the default PWM\_SYNC width is 8.19  $\mu$ s, for a 125 MHz  $f_{CLK}$ .

## External PWM SYNC Generation

By setting the PWM\_CTL.EXTSYNC bit, the PWM is set up in a mode to expect an external PWM\_SYNC on the PWM\_SYNC\_IN pin. The external PWM\_SYNC only determines the operation of the main timer PWMTMR0.

The external sync should be synchronized by setting the PWM\_CTL.EXTSYNCSEL bit to 0 (assumes the external PWM\_SYNC selected is asynchronous).

The external PWM\_SYNC period is expected to be an integer multiple of the value of the PWM\_TMO period register. When the rising edge of the external PWM\_SYNC is detected, the PWMTMR0 timer is restarted at the beginning of its period. If the external PWM\_SYNC period is not exactly an integer multiple of the internal PWM\_SYNC, the behavior of the PWM channel outputs which are referenced to PWMTMR0 are clipped.

The effect latency from PWM\_SYNC\_IN to the PWM outputs is about three clock cycles in synchronous mode, and five clock cycles in asynchronous mode.

## Event Control

The PWM output signals can be shut-off in a number of different ways. Two trip inputs pwm\_trip0b, and pwm\_trip1b are provided, each of which can be mapped to provide either a temporary or permanent shut-down on any pair of channel outputs. This shutdown mechanism is asynchronous so that the associated PWM output disable circuitry does not go through any clocked logic, ensuring correct PWM shutdown

even in the event of a loss of the processor clock. In addition to the hardware shutdown features, the PWM system may be shutdown in software by means of the `PWM_CTL.SWTRIP` bit.

Status information about the PWM is available to the user in the `PWM_STAT` register, which stores all status bits, including raw interrupt status bits. In particular, the period boundary of each timer is available, as well as status bits that indicate whether the operation is in the first half or the second half of the timer. Additionally the TRIP status is also available.

The `PWM_IMSK` and `PWM_ILAT` registers allow masking and show masked interrupt status bits respectively. The interrupt bits are latched and held on the interrupt event and the software must write a 1 to clear the interrupt bit, usually during the Interrupt Service routine.

## Trip Control Unit

The PWM Trip unit processes hardware or software fault conditions and shuts down the PWM channel outputs immediately on the occurrence of these conditions. This shut down mechanism can be enabled separately for each channel. The design also allows for a self-restart mechanism to be enabled on a channel. Self-restart re-enables the channel outputs following the fault condition (allowed only on hardware trips) when the PWMTMR-y that the channel is using reaches its period boundary.

There are 2 external hardware sources that can indicate a hardware fault condition:

1. `PWMTRIP0` input pin
2. `PWMTRIP1` input pin

These are active low inputs where a falling edge on either of these pins indicates a fault condition.

To enable the trip unit to shut down a particular channel's output in response to the fault event on either of these `PWM_TRIPN` pins, program the `PWM_TRIPCFG.ENOA` bit corresponding to that channel.

The `PWM_TRIPCFG.MODE0A` bits must be programmed to specify the restart mechanism for a channel that has been tripped.

1. If `PWM_TRIPCFG.MODE0A = 0`, once tripped, a trip condition is registered on this channel in the `PWM_STAT.FLTTRIPA` bit and the outputs of that channel are immediately shut down. This is called a **FAULT TRIP** condition. To resume channel output when a FAULT TRIP occurs, clear the `PWM_STAT.FLTTRIPA` bit by writing a 1 to it. Note that the bit cannot be cleared by a processor write if the trip condition is still active. The RAW trip status is available for both the pins in the `PWM_STAT.RAWTRIP0` register bits.
2. If `PWM_TRIPCFG.MODE0A = 1`, once tripped, a trip condition is registered on this channel in the `PWM_STAT.SRTRIPA` bit and the outputs of that channel are immediately shut down. This is called a **self-restart trip** condition. At the next period boundary of the PWMTMR-y that the channel is using, if the tripping condition is not active, the design clears the status register bit and restores the outputs.

The trip input pins should have an external pull-down resistor on the chip pin, so that if the pin becomes unconnected the PWM will be disabled.

In addition to the hardware trip conditions, a global software trip bit in the `PWM_CTL` register allows for a software forced FAULT TRIP condition. When the global software trip bit is set to 1, irrespective of the values in the `PWM_TRIPCFG` register, it sets all the `PWM_STAT.FLTTRIPA` bits in the `PWM_STAT` register and also gates the channel outputs. To remove the trip condition from the channel, a W1C should be performed on the particular channel's `PWM_STAT.FLTTRIPA` bit.

If the `PWM_TRIPCFG.ENOA` bit is set to 1 to, for any channel, then the occurrence of a fault condition on the `PWMTRIPy` bit is logged in the `PWM_STAT.FLTTRIPA` register bit. If the corresponding `PWM_IMSK.TRIPO` bit = 1, then an interrupt is generated. Note that tripping a channel output doesn't interfere with `PWM_SYNC` generation.

The following figure shows a scenario where `PWMTRIP0` is enabled on channel A as SELF RESTART TRIP. Channel A works with `PWM_CHANCFG.POLAH = 1`. Note that in Period 2, `PWM_AH` is full ON modulated, and tries to rise at the period boundary itself where the self-restart occurs for the channel. However, since the low-side output of the channel was only recently removed due to a trip, the rise edge on `PWM_AH` is delayed until the emergency dead-time period is over. `PWMTRIP1` is enabled on channel B as FAULT TRIP. Channel B works with `PWM_CHANCFG.POLAH = 0`. `PWMTRIP1` stays low for a long time, and because of this the first processor write to re-enable the channel output fails. The second one passes since the fault-condition has gone away.

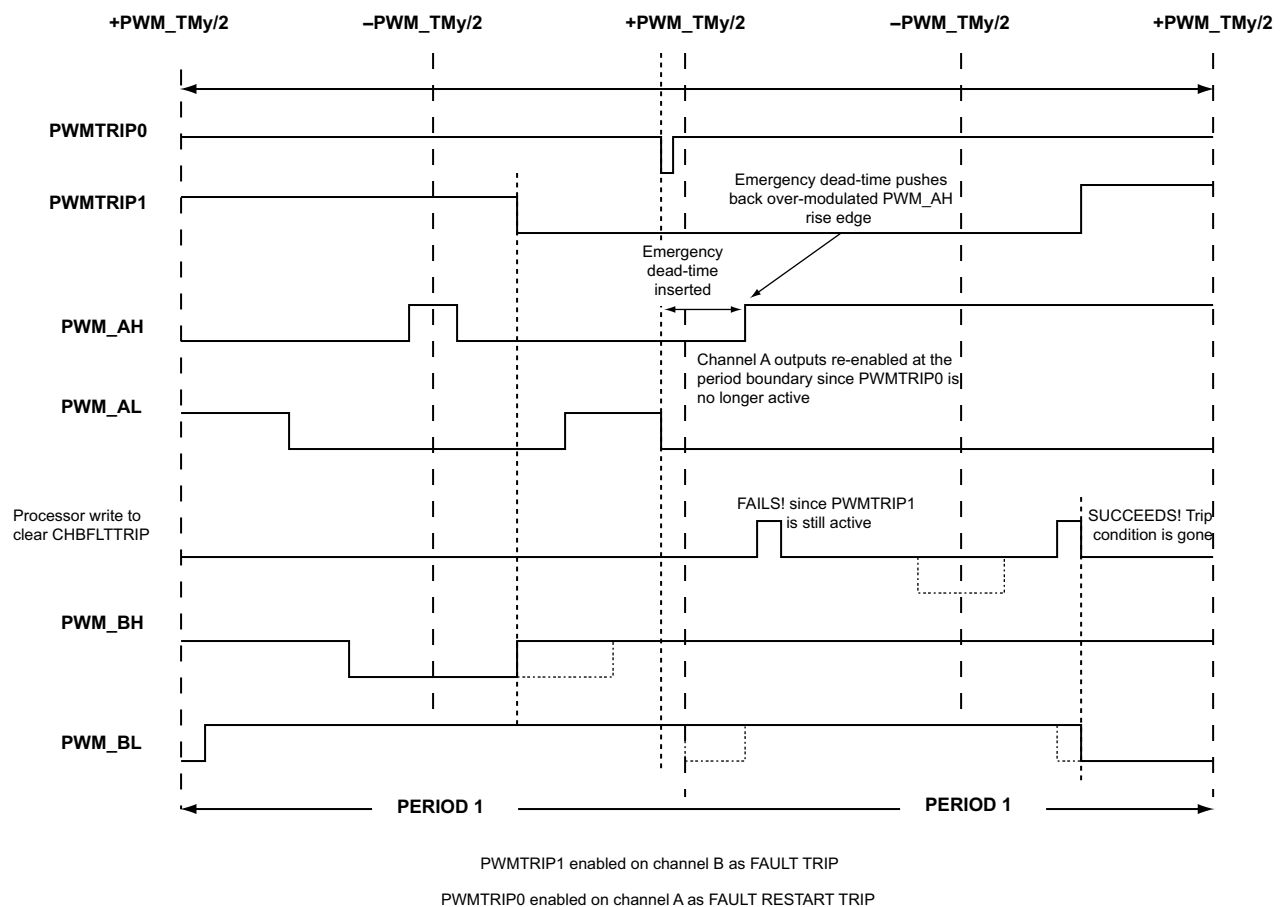


**Figure 18-18:** Operation Under Hardware Fault Conditions

**NOTE:**

Dead-time is ensured on re-enabling the channel outputs after trip.

**NOTE:**

Programs should not allow changes in the configuration/enable bits of PWM\_TRIPCFG register (which select between trip enable and disable) within  $\pm 10$  clock cycles of the toggling of external trip pulse. If this time frame is not followed, then unexpected behavior occurs.

## Programming Model

The following sections provide general (and some application specific) programming steps for configuring and using the PWM module.

- [Programming Model for 3-Phase AC Motor Control](#)

## Programming Model for 3-Phase AC Motor Control

The [PWM Module and Interaction with System](#) figure shows how the PWM unit (green) interfaces to both software (blue) and hardware (yellow). The software configures the unit, calculates duty cycles (Duty A, Duty B, Duty C), and services the interrupts generated by the module (PWM Sync IRQ, TRIP IRQ). The hardware applies the gate signals (AH, AL, BH, BL, CH, CL) to the inverter and provides an overcurrent trip signal back to the unit (TRIP0).

The typical 3-phase AC motor configuration shown in the [PWM Module and Interaction with System](#) figure applies for both permanent magnet and induction motor types.

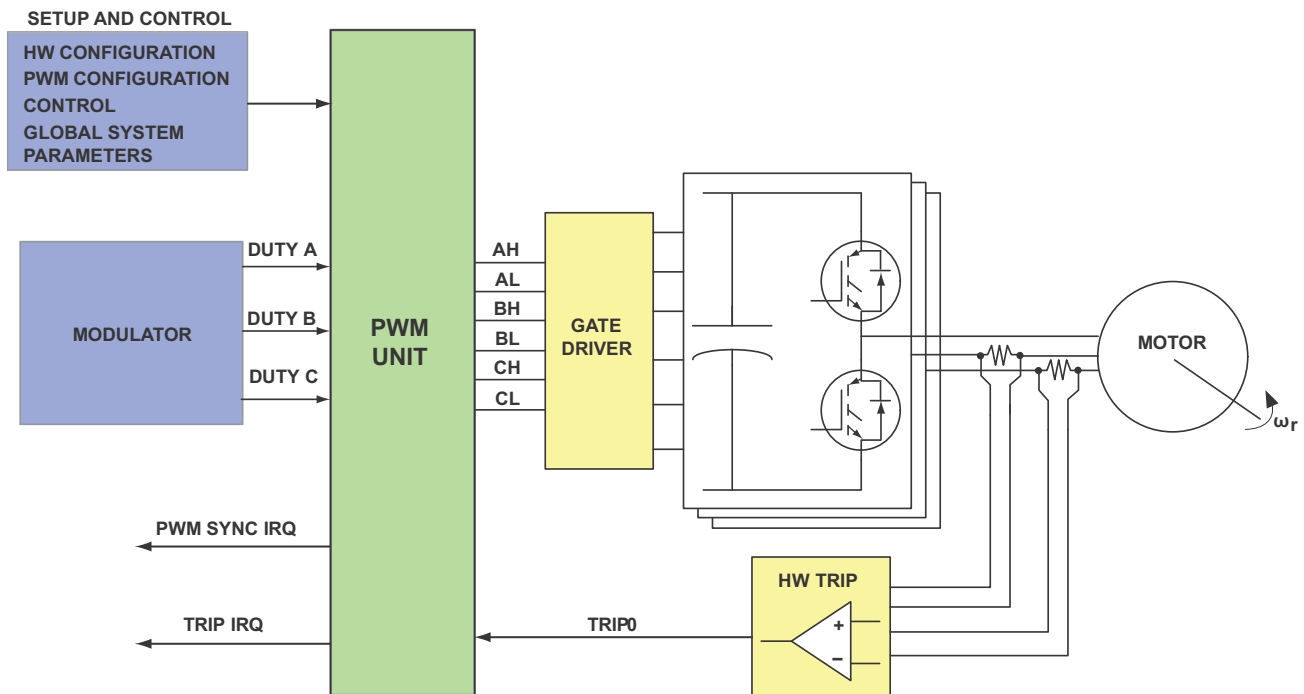


**Figure 18-19:** PWM Module and Interaction with System

## System Parameters

The following system parameters (characteristics) influence the module configuration for this application. This example system has/uses:

- One 3-phase AC machine
- B6 inverter
- SVPWM, including both linear- and over-modulation
- Switching frequency of 20 kHz
- Dead time of 1us
- Trip signal generated by hardware
- Active high level gate drive
- Core frequency of 200 MHz
- Peripheral clock of 100 MHz

## System State Sequencing

Managing the system state and sequence of states is critically important when programming the PWM module. The *PWM System States* figure provides an overview of these states.

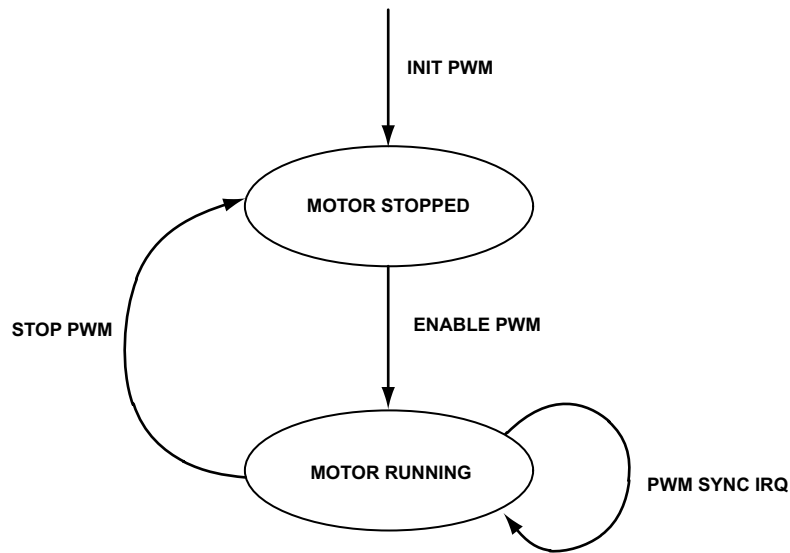


**Figure 18-20:** PWM System States

As shown in this state diagram, the module configuration is updated on state transitions (indicated by the arrows). The transitions are initialization, motor start, PWM sync interrupt (on each), and motor stop. These transitions are discussed in detail in the following sections.

- *PWM Initialization for Motor Control*
- *PWM Enable for Motor Control*
- *PWM Response to Sync Interrupt for Motor Control*
- *PWM Disable (and Stop the Motor) for Motor Control*

## PWM Initialization for Motor Control

The processor should do the following programming at power up and repeat this programming whenever the PWM and system must be brought into a known (safe) state.

1. Place the PWM module in a safe state and set up synchronization of the module.

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_CTL and PWM\_CHANCFG register accomplish this PWM state:

```
PWM_CTL &= 0xFFE0FF08  
PWM_CTL |= 0x20000
```

```
PWM_CHANCFG &= 0x80808080
PWM_CHANCFG |= 0x24242424
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Disable PWM ( PWM\_CTL.GLOBEN = 0)
- All phases must run with same phase, disable delay for channels A, B, C, D (PWM\_CTL.DLYAEN through PWM\_CTL.DLYDEN =0)
- Use internal synchronization by timer TMR0 (PWM\_CTL.EXTSYNC =0, PWM\_CTL.EXTSYNCSEL =1)
- All phases must be synchronized by the same timer, TMR0 (PWM\_CTL.INTSYNCREF = b#000)
- Low side is always the inverse of High side (PWM\_CHANCFG.POLAL through PWM\_CHANCFG.POLDL = 1)
- System uses active high gate driver (PWM\_CHANCFG.ENCHOPAH through PWM\_CHANCFG.ENCHOPDH =1)
- Pulse transformer is not used: disable gate chopping (PWM\_CHANCFG.ENCHOPAL through PWM\_CHANCFG.ENCHOPDL =0)

## 2. Set up the trip and associated interrupts.

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_TRIPCFG and PWM\_ILAT register accomplish this PWM state:

```
PWM_TRIPCFG &= 0xF0F0F0F0
PWM_TRIPCFG |= 0x1010101
PWM_ILAT &= 0xFFE0FFFC
PWM_ILAT |= 0x1
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- All phases must shut down simultaneously in case of fault: (PWM\_TRIPCFG.ENOA through PWM\_TRIPCFG.ENOD =0, PWM\_TRIPCFG.MODEOA through PWM\_TRIPCFG.MODEOD =0,

PWM\_TRIPCFG.EN1A through PWM\_TRIPCFG.EN1D =0, PWM\_TRIPCFG.MODE1A through PWM\_TRIPCFG.MODE1D =0)

- Enable TRIP0 as fault trigger for all channels. (PWM\_TRIPCFG.EN0A through PWM\_TRIPCFG.MODE1D =1)
  - For thermal control and synchronization, SW intervention is needed at trip. Do not use automatic restart of any channels
  - Generate an interrupt at trip on TRIP0. (PWM\_ILAT.TMROPER = 1)
3. Configure the PWM channels.

*ADDITIONAL INFORMATION:*

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_TRIPCFG and PWM\_ILAT register accomplish this PWM state:

```
PWM_DT = 0x32
PWM_TMO = 0x9C4
PWM_ACTL &= 0xFFFFF0000
PWM_BCTL &= 0xFFFFF0000
PWM_CCTL& = 0xFFFFF0000
PWM_AH0 = 0x0
```



```
PWM_BH0 = 0x0
PWM_CH0 = 0x0
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Configure a dead time of 1  $\mu$ s (PWM\_DT = 0x32).
- Configure a PWM frequency of 20 kHz (PWM\_TMO = 0x9C4).
- Disable all outputs (PWM\_ACTL.DISHI- PWM\_CCTL.DISHI = 0, PWM\_ACTL.DISLO through PWM\_CCTL.DISLO = 0)
- Use conventional PWM, disable crossover (PWM\_ACTL.XOVR through PWM\_CCTL.XOVR = 0)
- Use symmetrical pulse position on all outputs (PWM\_ACTL.PULSEMODEHI through PWM\_CCTL.PULSEMODEHI = 0, PWM\_ACTL.PULSEMODELO through PWM\_CCTL.PULSEMODELO = 0)
- Set an initial duty-cycle of 50% (PWM\_AH0 through PWM\_CH0 = 0x0)

## PWM Enable for Motor Control

The processor must do the following programming to enable the PWM before starting the motor.

1. Start the PWM timer TMR0.

*ADDITIONAL INFORMATION:* The following bitwise operation on the PWM\_CTL register accomplishes this PWM state:

```
PWM_CTL |= 0x1
```

*ADDITIONAL INFORMATION:* This operation achieves the following bit setting:

- Enable PWM ( PWM\_CTL.GLOBEN =1)

2. Enable six PWM outputs.

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_ACTL through PWM\_CCTL registers accomplish this PWM state:

```
PWM_ACTL |= 0x3
PWM_BCTL |= 0x3
PWM_CCTL |= 0x3
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Enable high and low side channel outputs (PWM\_ACTL.DISHI through PWM\_CCTL.DISHI = 1, PWM\_ACTL.DISLO through PWM\_CCTL.DISLO = 1)

3. Enable the PWM TRIP0 interrupt.

*ADDITIONAL INFORMATION:* The following bitwise operation on the PWM\_ILAT register accomplishes this PWM state:

```
PWM_ILAT |= 0x1
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Enable PWM TRIP0 interrupt ( `PWM_ILAT.TRIPO =1`)

## **PWM Response to Sync Interrupt for Motor Control**

When the PWM sync interrupt occurs, the processor may need to update to the PWM duty cycle with a value calculated by the motor control algorithm. This application uses symmetric pulses position and uses dependent High and Low side outputs, so only one register needs to be updated for each phase.

1. Write new duty cycle value (calculated by motor control algorithm) to the timer when the sync interrupt occurs.

*ADDITIONAL INFORMATION:* The following bitwise operations on the `PWM_AH0` through `PWM_CH0` registers accomplish this PWM state:

```
PWM_AH0 = Duty_A_mc_algorithm_current_value
```

*ADDITIONAL INFORMATION:*

```
PWM_BH0 = Duty_B_mc_algorithm_current_value
```

*ADDITIONAL INFORMATION:*

```
PWM_CH0 = Duty_C_mc_algorithm_current_value
```

*ADDITIONAL INFORMATION:*

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

## PWM Disable (and Stop the Motor) for Motor Control

The processor should do the following programming to stop the motor, disable the PWM, and disable PWM interrupts. These actions place the PWM and system in a safe, passive state.

1. Disable the PWM timer.

*ADDITIONAL INFORMATION:* The following bitwise operation on the PWM\_CTL register accomplish this PWM state:

```
PWM_CTL &= 0xFFFFFFFF
```

*ADDITIONAL INFORMATION:* This operation achieves the following bit setting:

- Disable PWM (PWM\_CTL.GLOBEN = 0)

2. Disable all PWM outputs.

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_ACTL through PWM\_CCTL registers accomplish this PWM state:

```
PWM_ACTL &= 0xFFFFFFFFC
```

```
PWM_BCTL &= 0xFFFFFFFFC
```

```
PWM_CCTL &= 0xFFFFFFFFC
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Disable PWM outputs PWM\_ACTL.DISHI through PWM\_CCTL.DISHI = 1, PWM\_ACTL.DISLO through PWM\_CCTL.DISLO = 0)

3. Set the PWM duty-cycle to 50%.

*ADDITIONAL INFORMATION:* The following bitwise operations on the PWM\_AHO through PWM\_CHO registers accomplish this PWM state:

```
PWM_AHO = 0x0
```

```
PWM_BHO = 0x0
```

```
PWM_CHO = 0x0
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Set PWM duty cycle to 50%. (PWM\_AHO.DUTY through PWM\_CHO.DUTY = 0)

4. Disable the PWM TRIP0 interrupt.

*ADDITIONAL INFORMATION:* The following bitwise operation on the PWM\_ILAT register accomplish this PWM state:

```
PWM_ILAT &= 0xFFFFFFFF
```

*ADDITIONAL INFORMATION:* These operations achieve the following bit settings:

- Disable PWM TRIP0 interrupt (PWM\_ILAT.TRIPO = 0)

## PWM Register Descriptions

Pulse-Width Modulator (PWM) contains the following registers.

**Table 18-5:** PWM Module Registers

| Name         | Description                    |
|--------------|--------------------------------|
| PWM_CTL      | Control Register               |
| PWM_CHANCFG  | Channel Config Register        |
| PWM_TRIPCFG  | Trip Config Register           |
| PWM_STAT     | Status Register                |
| PWM_IMSK     | Interrupt Mask Register        |
| PWM_ILAT     | Interrupt Latch Register       |
| PWM_CHOPCFG  | Chop Configuration Register    |
| PWM_DT       | Dead Time Register             |
| PWM_SYNC_WID | Sync Pulse Width Register      |
| PWM_TM0      | Timer 0 Period Register        |
| PWM_TM1      | Timer 1 Period Register        |
| PWM_TM2      | Timer 2 Period Register        |
| PWM_TM3      | Timer 3 Period Register        |
| PWM_TM4      | Timer 4 Period Register        |
| PWM_DLYA     | Channel A Delay Register       |
| PWM_DLYB     | Channel B Delay Register       |
| PWM_DLYC     | Channel C Delay Register       |
| PWM_DLYD     | Channel D Delay Register       |
| PWM_ACTL     | Channel A Control Register     |
| PWM_AH0      | Channel A-High Duty-0 Register |

**Table 18-5:** PWM Module Registers (Continued)

| Name     | Description                          |
|----------|--------------------------------------|
| PWM_AH1  | Channel A-High Duty-1 Register       |
| PWM_AL0  | Channel A-Low Duty-0 Register        |
| PWM_AL1  | Channel A-Low Duty-1 Register        |
| PWM_BCTL | Channel B Control Register           |
| PWM_BH0  | Channel B-High Duty-0 Register       |
| PWM_BH1  | Channel B-High Duty-1 Register       |
| PWM_BL0  | Channel B-Low Duty-0 Register        |
| PWM_BL1  | Channel B-Low Duty-1 Register        |
| PWM_CCTL | Channel C Control Register           |
| PWM_CH0  | Channel C-High Pulse Duty Register 0 |
| PWM_CH1  | Channel C-High Pulse Duty Register 1 |
| PWM_CL0  | Channel C-Low Pulse Duty Register 0  |
| PWM_CL1  | Channel C-Low Duty-1 Register        |
| PWM_DCTL | Channel D Control Register           |
| PWM_DH0  | Channel D-High Duty-0 Register       |
| PWM_DH1  | Channel D-High Pulse Duty Register 1 |
| PWM_DL0  | Channel D-Low Pulse Duty Register 0  |
| PWM_DL1  | Channel D-Low Pulse Duty Register 1  |

## Control Register

The PWM\_CTL register enables the PWM, enables delay counters for the channels, and configures sync features. This register also provides support for tripping a PWM fault condition through software.

PWM\_CTL: Control Register - R/W

Reset = 0x0002 0000

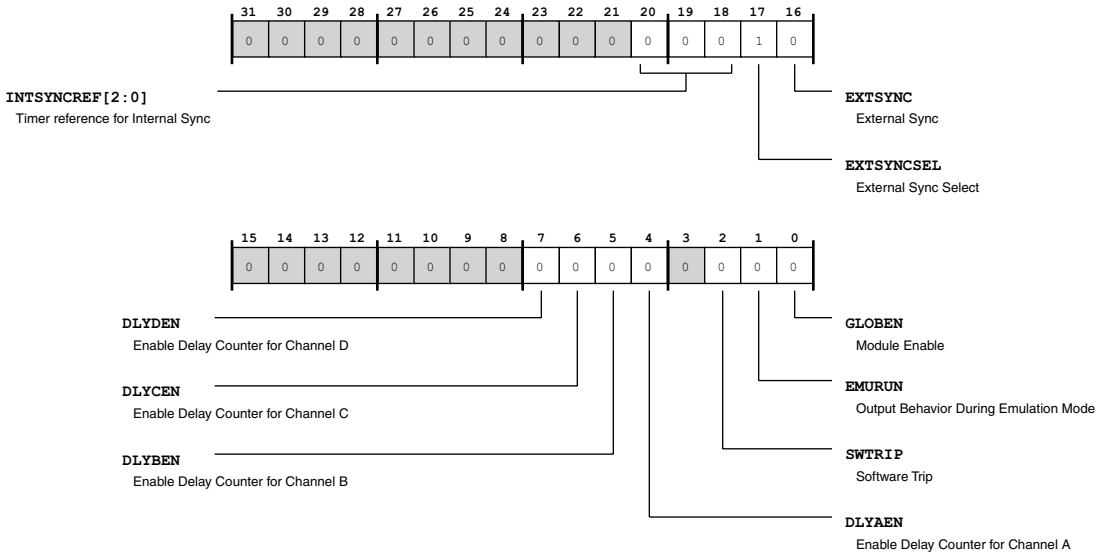


Figure 18-21: PWM\_CTL Register Diagram

Table 18-6: PWM\_CTL Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                          |
|---------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20:18   | INTSYNCREF | Timer reference for Internal Sync.<br>The PWM_CTL . INTSYNCREF bits select the timer reference for the internal sync. Note that all other combinations reserved. |
|         |            | 0 PWMTMR0 provides sync reference                                                                                                                                |
|         |            | 1 PWMTMR1 provides sync reference                                                                                                                                |
|         |            | 2 PWMTMR2 provides sync reference                                                                                                                                |
|         |            | 3 PWMTMR3 provides sync reference                                                                                                                                |
|         |            | 4 PWMTMR4 provides sync reference                                                                                                                                |

**Table 18-6:** PWM\_CTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                          |
|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | EXTSYNCSSEL | External Sync Select.<br>The PWM_CTL.EXTSYNCSSEL bit selects whether the external sync signal is synchronous or asynchronous. Note that latency in PWM sync response differs between asynchronous and synchronous external sync modes. For more information, see the PWM functional description. |
|         |             | 0 Asynchronous External Sync                                                                                                                                                                                                                                                                     |
|         |             | 1 Synchronous External Sync                                                                                                                                                                                                                                                                      |
| 16      | EXTSYNC     | External Sync.<br>The PWM_CTL.EXTSYNC bit selects whether the PWM uses an external or internal sync signal for the main timer (PWMTMR0). Do not change the value of the PWM_CTL.EXTSYNC bit while the PWM is enabled (PWM_CTL.GLOBEN =1).                                                        |
|         |             | 0 Internal sync used                                                                                                                                                                                                                                                                             |
|         |             | 1 External sync used                                                                                                                                                                                                                                                                             |
| 7       | DLYDEN      | Enable Delay Counter for Channel D.<br>The PWM_CTL.DLYDEN bit enables the Channel D delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYDEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).                                                              |
|         |             | 0 Disable                                                                                                                                                                                                                                                                                        |
|         |             | 1 Enable                                                                                                                                                                                                                                                                                         |
| 6       | DLYCEN      | Enable Delay Counter for Channel C.<br>The PWM_CTL.DLYCEN bit enables the Channel C delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYCEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).                                                              |
|         |             | 0 Disable                                                                                                                                                                                                                                                                                        |
|         |             | 1 Enable                                                                                                                                                                                                                                                                                         |

**Table 18-6:** PWM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | DLYBEN   | Enable Delay Counter for Channel B.<br>The PWM_CTL.DLYBEN bit enables the Channel B delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYBEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).                                         |
|         |          | 0      Disable                                                                                                                                                                                                                                                              |
|         |          | 1      Enable                                                                                                                                                                                                                                                               |
| 4       | DLYAEN   | Enable Delay Counter for Channel A.<br>The PWM_CTL.DLYAEN bit enables the Channel A delay counter, supporting phase-offset control. Do not change the value of the PWM_CTL.DLYAEN bit while the PWM is enabled (PWM_CTL.GLOBEN =1).                                         |
|         |          | 0      Disable                                                                                                                                                                                                                                                              |
|         |          | 1      Enable                                                                                                                                                                                                                                                               |
| 2       | SWTRIP   | Software Trip.<br>The PWM_CTL.SWTRIP bit permits tripping a fault condition through software, shutting down PWM output. This bit always read as 0. If the PWM_CTL.SWTRIP bit and PWM_CTL.GLOBEN bit are set in the same write, the write does not trip the fault condition. |
|         |          | 1      Force a Fault Trip Condition                                                                                                                                                                                                                                         |
| 1       | EMURUN   | Output Behavior During Emulation Mode.<br>The PWM_CTL.EMURUN bit selects PWM output behavior during emulation mode.                                                                                                                                                         |
|         |          | 0      Disable Outputs                                                                                                                                                                                                                                                      |
|         |          | 1      Enable Outputs                                                                                                                                                                                                                                                       |



**Table 18-6:** PWM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | GLOBEN   | Module Enable.<br>The PWM_CTL.GLOBEN bit enables the PWM, enabling all timers and outputs. While this bit is enabled, processor code should not change the value of the PWM_CTL.DLYAEN bit, PWM_CTL.DLYBEN bit, PWM_CTL.DLYCEN bit, PWM_CTL.DLYDEN bit, PWM_CTL.EXTSYNCSEL bit, or any bits in the PWM_CHANCFG register. Note that there is a latency between PWM disable and the cessation of output waveforms. There is also a latency between PWM enable and start of output waveforms. For the latency description, see the PWM functional description. |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## Channel Config Register

The PWM\_CHANCFG register configures Channel A, B, C, and D reference timer selection, high and low side output features, and enables high frequency chopping operation. Do not change the value of any bits in the PWM register while the PWM is enabled (PWM\_CTL.GLOBEN =1).

PWM\_CHANCFG: Channel Config Register - R/W

Reset = 0x0000 0000

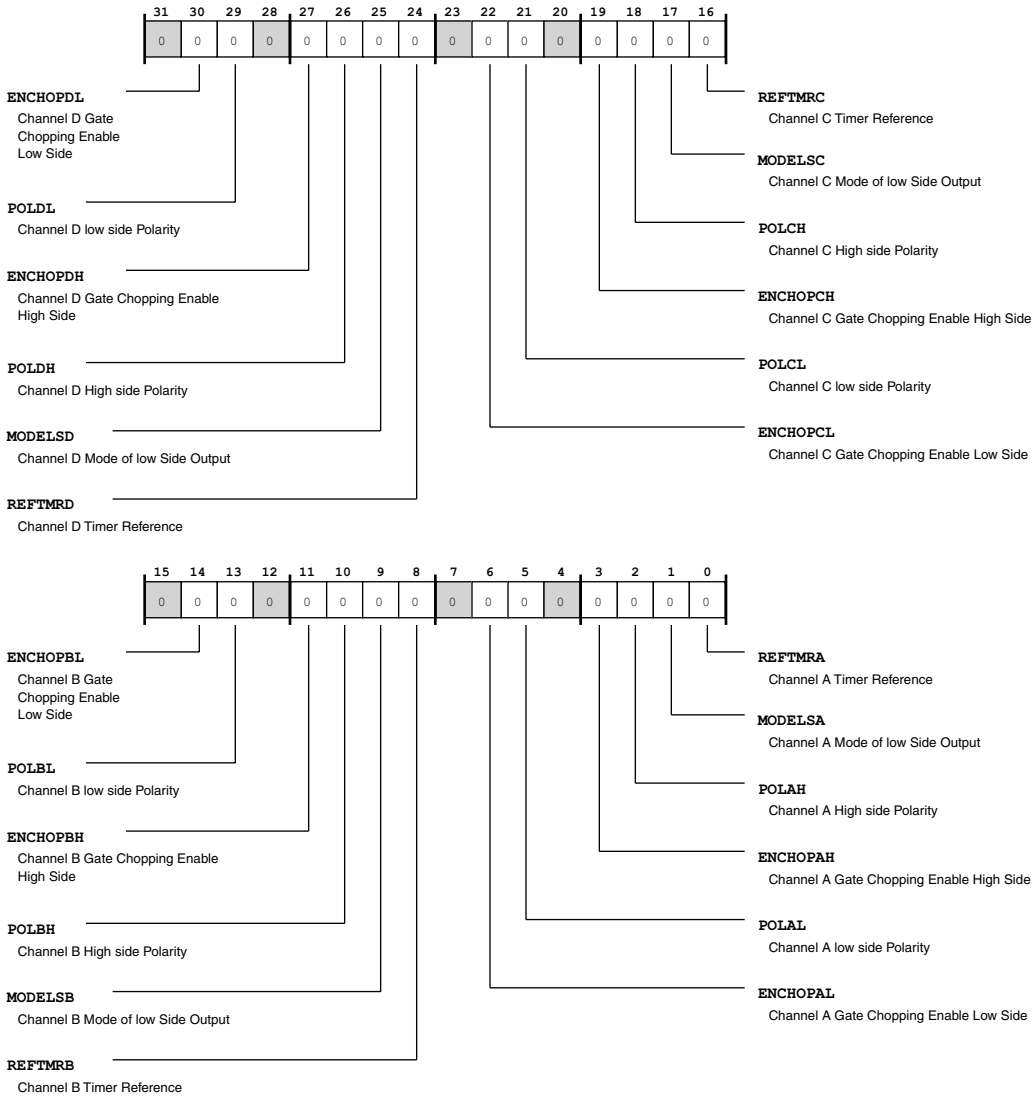


Figure 18-22: PWM\_CHANCFG Register Diagram

**Table 18-7:** PWM\_CHANCFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30      | ENCHOPDL | Channel D Gate Chopping Enable Low Side.<br>The PWM_CHANCFG.ENCHOPDL bit enables mixing of the Channel D low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.   |
|         |          | 0      Disable Chopping Channel D Low Side                                                                                                                                                                                   |
|         |          | 1      Enable Chopping Channel D Low Side                                                                                                                                                                                    |
| 29      | POLDL    | Channel D low side Polarity.<br>The PWM_CHANCFG.POLDL bit selects the Channel D low side output polarity (active-high or active-low).                                                                                        |
|         |          | 0      Active Low                                                                                                                                                                                                            |
|         |          | 1      Active High                                                                                                                                                                                                           |
| 27      | ENCHOPDH | Channel D Gate Chopping Enable High Side.<br>The PWM_CHANCFG.ENCHOPDH bit enables mixing of the Channel D high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register. |
|         |          | 0      Disable Chopping Channel D High Side                                                                                                                                                                                  |
|         |          | 1      Enable Chopping Channel D High Side                                                                                                                                                                                   |
| 26      | POLDH    | Channel D High side Polarity.<br>The PWM_CHANCFG.POLDH bit selects the Channel D high side output polarity (active-high or active-low).                                                                                      |
|         |          | 0      Active Low                                                                                                                                                                                                            |
|         |          | 1      Active High                                                                                                                                                                                                           |

**Table 18-7:** PWM\_CHANCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | MODELSD  | Channel D Mode of low Side Output.<br>The PWM_CHANCFG.MODELSD bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSD =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity. |
|         |          | 0      Invert of high output                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1      Independent control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 24      | REFTMRD  | Channel D Timer Reference.<br>The PWM_CHANCFG.REFTMRD bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel D operation.                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      PWMTMR0 is Channel D reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1      PWMTMR1 is Channel D reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 22      | ENCHOPCL | Channel C Gate Chopping Enable Low Side.<br>The PWM_CHANCFG.ENCHOPCL bit enables mixing of the Channel C low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.                                                                                                                                                                                                                                                                             |
|         |          | 0      Disable Chopping Channel C Low Side                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1      Enable Chopping Channel C Low Side                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 21      | POLCL    | Channel C low side Polarity.<br>The PWM_CHANCFG.POLCL bit selects the Channel C low side output polarity (active-high or active-low).                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 0      Active Low                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Active High                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**Table 18-7:** PWM\_CHANCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19      | ENCHOPCH | Channel C Gate Chopping Enable High Side.<br>The PWM_CHANCFG.ENCHOPCH bit enables mixing of the Channel C high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.                                                                                                                                                                                                                                                                           |
|         |          | 0      Disable Chopping Channel C High Side                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Enable Chopping Channel C High Side                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 18      | POLCH    | Channel C High side Polarity.<br>The PWM_CHANCFG.POLCH bit selects the Channel C high side output polarity (active-high or active-low).                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 0      Active Low                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Active High                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 17      | MODELSC  | Channel C Mode of low Side Output.<br>The PWM_CHANCFG.MODELSC bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSC =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity. |
|         |          | 0      Invert of high output                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1      Independent control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 16      | REFTMRC  | Channel C Timer Reference.<br>The PWM_CHANCFG.REFTMRC bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel C operation.                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      PWMTMR0 is Channel C reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1      PWMTMR1 is Channel C reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Table 18-7:** PWM\_CHANCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | ENCHOPBL | Channel B Gate Chopping Enable Low Side.<br>The PWM_CHANCFG.ENCHOPBL bit enables mixing of the Channel B low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.   |
|         |          | 0      Disable Chopping Channel B Low Side                                                                                                                                                                                   |
|         |          | 1      Enable Chopping Channel B Low Side                                                                                                                                                                                    |
| 13      | POLBL    | Channel B low side Polarity.<br>The PWM_CHANCFG.POLBL bit selects the Channel B low side output polarity (active-high or active-low).                                                                                        |
|         |          | 0      Active Low                                                                                                                                                                                                            |
|         |          | 1      Active High                                                                                                                                                                                                           |
| 11      | ENCHOPBH | Channel B Gate Chopping Enable High Side.<br>The PWM_CHANCFG.ENCHOPBH bit enables mixing of the Channel B high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register. |
|         |          | 0      Disable Chopping Channel B High Side                                                                                                                                                                                  |
|         |          | 1      Enable Chopping Channel B High Side                                                                                                                                                                                   |
| 10      | POLBH    | Channel B High side Polarity.<br>The PWM_CHANCFG.POLBH bit selects the Channel B high side output polarity (active-high or active-low).                                                                                      |
|         |          | 0      Active Low                                                                                                                                                                                                            |
|         |          | 1      Active High                                                                                                                                                                                                           |

**Table 18-7:** PWM\_CHANCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | MODELSB  | Channel B Mode of low Side Output.<br>The PWM_CHANCFG.MODELSB bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSB =0, the low side output is an inverted form of the high side output, which is generated using the PWM_BH0 and PWM_BH1 registers for pulse width, using the PWM_BCTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLBH bits for polarity. |
|         |          | 0      Invert of high output                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1      Independent control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 8       | REFTMRB  | Channel B Timer Reference.<br>The PWM_CHANCFG.REFTMRB bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel B operation.                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      PWMTMR0 is Channel B reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1      PWMTMR1 is Channel B reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 6       | ENCHOPAL | Channel A Gate Chopping Enable Low Side.<br>The PWM_CHANCFG.ENCHOPAL bit enables mixing of the Channel A low side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.                                                                                                                                                                                                                                                                             |
|         |          | 0      Disable Chopping Channel A Low Side                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1      Enable Chopping Channel A Low Side                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 5       | POLAL    | Channel A low side Polarity.<br>The PWM_CHANCFG.POLAL bit selects the Channel A low side output polarity (active-high or active-low).                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 0      Active Low                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Active High                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**Table 18-7:** PWM\_CHANCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | ENCHOPAH | Channel A Gate Chopping Enable High Side.<br>The PWM_CHANCFG.ENCHOPAH bit enables mixing of the Channel A high side output signals with a high-frequency chopping signal, which is configured with the PWM_CHOPCFG register.                                                                                                                                                                                                                                                                           |
|         |          | 0      Disable Chopping Channel A High Side                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Enable Chopping Channel A High Side                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 2       | POLAH    | Channel A High side Polarity.<br>The PWM_CHANCFG.POLAH bit selects the Channel A high side output polarity (active-high or active-low).                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 0      Active Low                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Active High                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 1       | MODELSA  | Channel A Mode of low Side Output.<br>The PWM_CHANCFG.MODELSA bit selects whether the low side output waveform is based on independent controls or whether the low side output depends on the high side output controls. When PWM_CHANCFG.MODELSA =0, the low side output is an inverted form of the high side output, which is generated using the PWM_AH0 and PWM_AH1 registers for pulse width, using the PWM_ACTL.PULSEMODEHI bits for pulse positioning, and PWM_CHANCFG.POLAH bits for polarity. |
|         |          | 0      Invert of high output                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1      Independent control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 0       | REFTMRA  | Channel A Timer Reference.<br>The PWM_CHANCFG.REFTMRA bit selects whether the PWM uses PWMTMR1 or PWMTMR0 as the reference timer for Channel A operation.                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      PWMTMR0 is Channel A reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1      PWMTMR1 is Channel A reference                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |



## Trip Config Register

The PWM\_TRIPCFG register configures Channel A, B, C, and D trip operation for trip inputs TRIP0 and TRIP1.

**PWM\_TRIPCFG: Trip Config Register - R/W**

Reset = 0x0000 0000

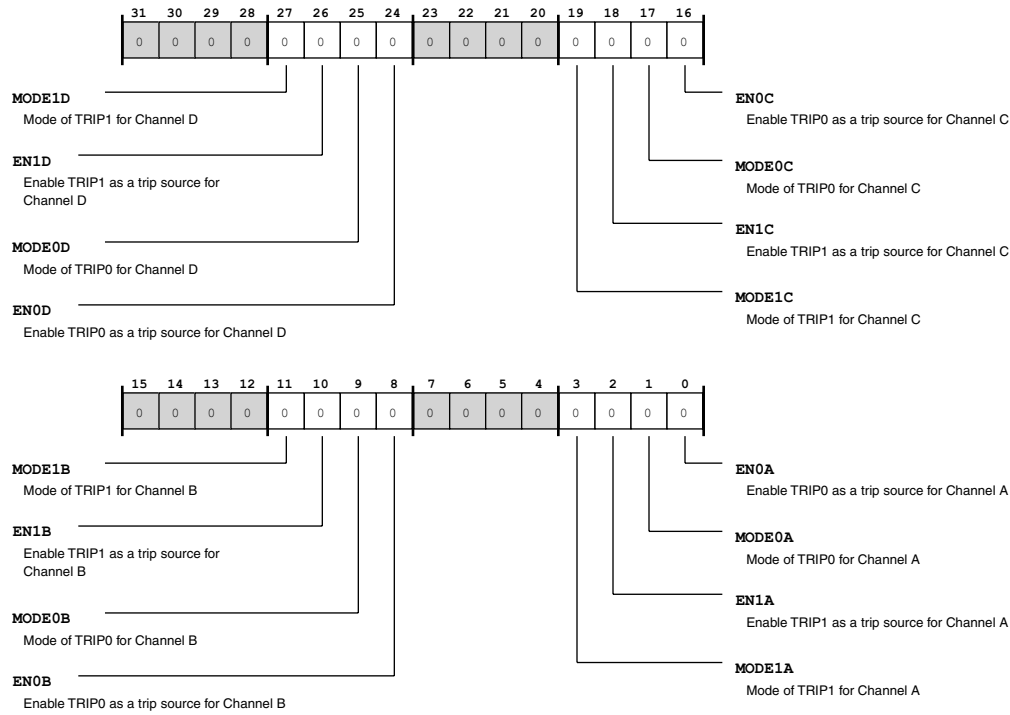


**Figure 18-23: PWM\_TRIPCFG Register Diagram**

**Table 18-8: PWM\_TRIPCFG Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27      | MODE1D   | Mode of TRIP1 for Channel D.<br>The PWM_TRIPCFG.MODE1D bit selects the trip mode of TRIP1 for Channel D. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0 Fault Trip on TRIP1 Input                                                                                                                                                |
|         |          | 1 Self Restart on TRIP1 Input                                                                                                                                              |

**Table 18-8:** PWM\_TRIPCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | EN1D     | Enable TRIP1 as a trip source for Channel D.<br>The PWM_TRIPCFG.EN1D bit enables TRIP1 as a trip source for Channel D.                                                     |
|         |          | 0      Disable TRIP1 for Channel D                                                                                                                                         |
|         |          | 1      Enable TRIP1 for Channel D                                                                                                                                          |
| 25      | MODE0D   | Mode of TRIP0 for Channel D.<br>The PWM_TRIPCFG.MODE0D bit selects the trip mode of TRIP0 for Channel D. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP0 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP0 Input                                                                                                                                         |
| 24      | EN0D     | Enable TRIP0 as a trip source for Channel D.<br>The PWM_TRIPCFG.EN0D bit enables TRIP0 as a trip source for Channel D.                                                     |
|         |          | 0      Disable TRIP0 for Channel D                                                                                                                                         |
|         |          | 1      Enable TRIP0 for Channel D                                                                                                                                          |
| 19      | MODE1C   | Mode of TRIP1 for Channel C.<br>The PWM_TRIPCFG.MODE1C bit selects the trip mode of TRIP1 for Channel C. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP1 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP1 Input                                                                                                                                         |
| 18      | EN1C     | Enable TRIP1 as a trip source for Channel C.<br>The PWM_TRIPCFG.EN1C bit enables TRIP1 as a trip source for Channel C.                                                     |
|         |          | 0      Disable TRIP1 for Channel C                                                                                                                                         |
|         |          | 1      Enable TRIP1 for Channel C                                                                                                                                          |

**Table 18-8:** PWM\_TRIPCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | MODE0C   | Mode of TRIP0 for Channel C.<br>The PWM_TRIPCFG.MODE0C bit selects the trip mode of TRIP0 for Channel C. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP0 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP0 Input                                                                                                                                         |
| 16      | EN0C     | Enable TRIP0 as a trip source for Channel C.<br>The PWM_TRIPCFG.EN0C bit enables TRIP0 as a trip source for Channel C.                                                     |
|         |          | 0      Disable TRIP0 for Channel C                                                                                                                                         |
|         |          | 1      Enable TRIP0 for Channel C                                                                                                                                          |
| 11      | MODE1B   | Mode of TRIP1 for Channel B.<br>The PWM_TRIPCFG.MODE1B bit selects the trip mode of TRIP1 for Channel B. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP1 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP1 Input                                                                                                                                         |
| 10      | EN1B     | Enable TRIP1 as a trip source for Channel B.<br>The PWM_TRIPCFG.EN1B bit enables TRIP1 as a trip source for Channel B.                                                     |
|         |          | 0      Disable TRIP1 for Channel B                                                                                                                                         |
|         |          | 1      Enable TRIP1 for Channel B                                                                                                                                          |
| 9       | MODE0B   | Mode of TRIP0 for Channel B.<br>The PWM_TRIPCFG.MODE0B bit selects the trip mode of TRIP0 for Channel B. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP0 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP0 Input                                                                                                                                         |

**Table 18-8:** PWM\_TRIPCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | EN0B     | Enable TRIP0 as a trip source for Channel B.<br>The PWM_TRIPCFG.EN0B bit enables TRIP0 as a trip source for Channel B.                                                     |
|         |          | 0      Disable TRIP0 for Channel B                                                                                                                                         |
|         |          | 1      Enable TRIP0 for Channel B                                                                                                                                          |
| 3       | MODE1A   | Mode of TRIP1 for Channel A.<br>The PWM_TRIPCFG.MODE1A bit selects the trip mode of TRIP1 for Channel A. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0      Fault Trip on TRIP1 Input                                                                                                                                           |
|         |          | 1      Self Restart on TRIP1 Input                                                                                                                                         |
| 2       | EN1A     | Enable TRIP1 as a trip source for Channel A.<br>The PWM_TRIPCFG.EN1A bit enables TRIP1 as a trip source for Channel A.                                                     |
|         |          | 0      Disable TRIP1 for Channel A                                                                                                                                         |
|         |          | 1      Enable TRIP1 for Channel A                                                                                                                                          |

**Table 18-8:** PWM\_TRIPCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                             |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 1       | MODE0A   | <p>Mode of TRIP0 for Channel A.</p> <p>The PWM_TRIPCFG.MODE0A bit selects the trip mode of TRIP0 for Channel A.</p> <p>In fault-trip mode (PWM_TRIPCFG.MODE0A =0), after the input is tripped, the trip status appears in the corresponding channel's fault-trip status bit (for example, PWM_STAT.FLTTRIPA), and the PWM immediately shuts down outputs of that channel. After a fault trip occurs, when the trip condition is no longer active, the processor may cause channel outputs to resume by completing a write-1-to-clear the corresponding fault-trip status bit. The raw (input level) trip input state is available from the PWM_STAT.RAWTRIP0 and PWM_STAT.RAWTRIP0 bits.</p> <p>In self-restart mode (PWM_TRIPCFG.MODE0A =1), after the input is tripped, the trip status appears in the corresponding channel's self-restart status bit (for example, PWM_STAT.SRTRIPA), and the PWM immediately shuts down outputs of that channel. On the next timer period boundary (of the PWMTMRx used by that channel), if the trip condition is not active, the PWM clears the status and restarts the channel's output.</p> |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Fault Trip on TRIP0 Input   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Self Restart on TRIP0 Input |
| 0       | EN0A     | <p>Enable TRIP0 as a trip source for Channel A.</p> <p>The PWM_TRIPCFG.EN0A bit enables TRIP0 as a trip source for Channel A.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Disable TRIP0 for Channel A |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Enable TRIP0 for Channel A  |

## Status Register

The PWM\_STAT register indicates the PWM PWMTRIP1-0 fault and input level status, indicates the Channel A-D fault and self-restart status, and indicates the PWMTMR4-0 phase.

PWM\_STAT: Status Register - R/W

Reset = 0x0000 0000

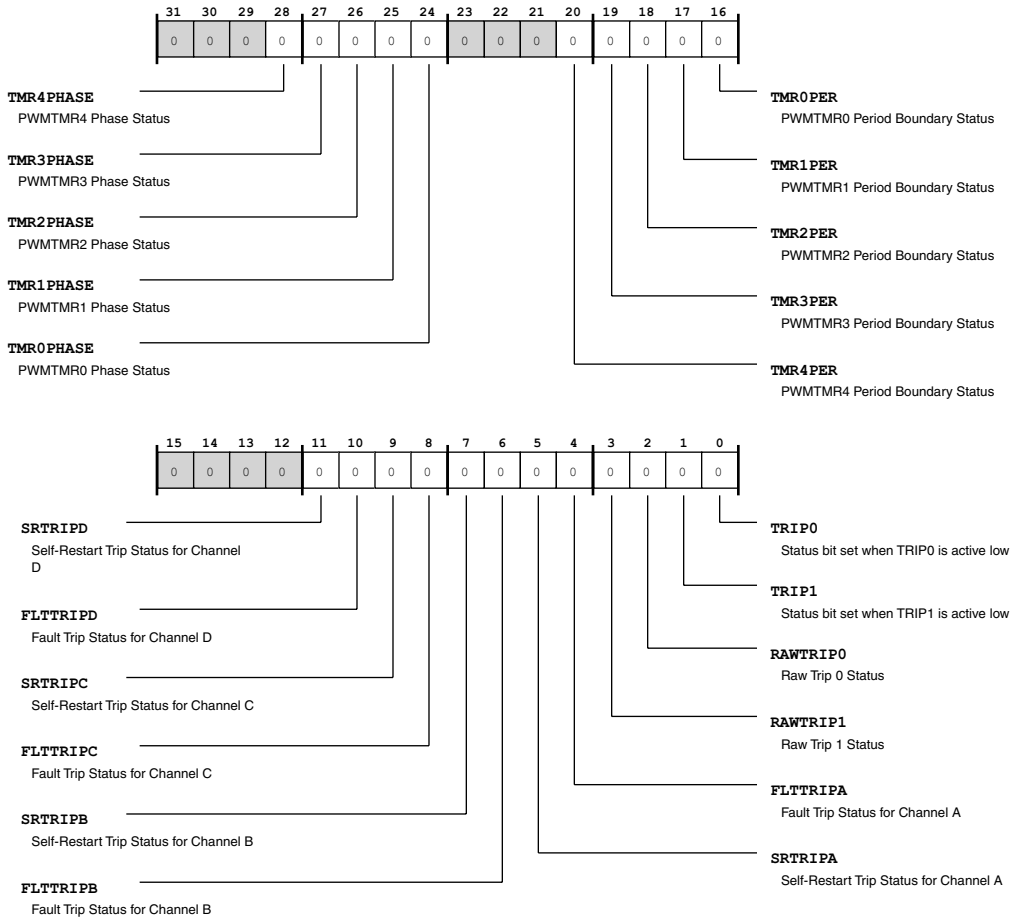


Figure 18-24: PWM\_STAT Register Diagram

Table 18-9: PWM\_STAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                   |
|---------|-----------|-----------------------------------------------------------------------------------------------------------|
| 28      | TMR4PHASE | PWMTMR4 Phase Status.<br>The PWM_STAT.TMR4PHASE bit indicates the current phase for the PWMTMR4 waveform. |
|         |           | 01st Half Phase                                                                                           |
|         |           | 12nd Half Phase                                                                                           |

**Table 18-9:** PWM\_STAT Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                            |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------|
| 27      | TMR3PHASE | PWMTMR3 Phase Status.<br>The PWM_STAT.TMR3PHASE bit indicates the current phase for the PWMTMR3 waveform.                          |
|         |           | 0 1st Half Phase                                                                                                                   |
|         |           | 1 2nd Half Phase                                                                                                                   |
| 26      | TMR2PHASE | PWMTMR2 Phase Status.<br>The PWM_STAT.TMR2PHASE bit indicates the current phase for the PWMTMR2 waveform.                          |
|         |           | 0 1st Half Phase                                                                                                                   |
|         |           | 1 2nd Half Phase                                                                                                                   |
| 25      | TMR1PHASE | PWMTMR1 Phase Status.<br>The PWM_STAT.TMR1PHASE bit indicates the current phase for the PWMTMR1 waveform.                          |
|         |           | 0 1st Half Phase                                                                                                                   |
|         |           | 1 2nd Half Phase                                                                                                                   |
| 24      | TMR0PHASE | PWMTMR0 Phase Status.<br>The PWM_STAT.TMR0PHASE bit indicates the current phase for the PWMTMR0 waveform.                          |
|         |           | 0 1st Half Phase                                                                                                                   |
|         |           | 1 2nd Half Phase                                                                                                                   |
| 20      | TMR4PER   | PWMTMR4 Period Boundary Status.<br>The PWM_STAT.TMR4PER bit indicates whether or not the PWMTMR4 period boundary has been reached. |
|         |           | 0 PWMTMR4 period boundary not reached                                                                                              |
|         |           | 1 PWMTMR4 period boundary reached                                                                                                  |

**Table 18-9:** PWM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19      | TMR3PER  | PWMTMR3 Period Boundary Status.<br>The PWM_STAT.TMR3PER bit indicates whether or not the PWMTMR3 period boundary has been reached.                                                                       |
|         |          | 0 PWMTMR3 period boundary not reached                                                                                                                                                                    |
|         |          | 1 PWMTMR3 period boundary reached                                                                                                                                                                        |
| 18      | TMR2PER  | PWMTMR2 Period Boundary Status.<br>The PWM_STAT.TMR2PER bit indicates whether or not the PWMTMR2 period boundary has been reached.                                                                       |
|         |          | 0 PWMTMR2 period boundary not reached                                                                                                                                                                    |
|         |          | 1 PWMTMR2 period boundary reached                                                                                                                                                                        |
| 17      | TMR1PER  | PWMTMR1 Period Boundary Status.<br>The PWM_STAT.TMR1PER bit indicates whether or not the PWMTMR1 period boundary has been reached.                                                                       |
|         |          | 0 PWMTMR1 period boundary not reached                                                                                                                                                                    |
|         |          | 1 PWMTMR1 period boundary reached                                                                                                                                                                        |
| 16      | TMR0PER  | PWMTMR0 Period Boundary Status.<br>The PWM_STAT.TMR0PER bit indicates whether or not the PWMTMR0 period boundary has been reached.                                                                       |
|         |          | 0 PWMTMR0 period boundary not reached                                                                                                                                                                    |
|         |          | 1 PWMTMR0 period boundary reached                                                                                                                                                                        |
| 11      | SRTRIPD  | Self-Restart Trip Status for Channel D.<br>The PWM_STAT.SRTRIPD bit indicates whether the PWM Channel D self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0 Channel D Self-Restart Trip Status is "not tripped"                                                                                                                                                    |
|         |          | 1 Channel D Self-Restart Trip Status is "tripped"                                                                                                                                                        |



**Table 18-9:** PWM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | FLTTRIPD | Fault Trip Status for Channel D.<br>The PWM_STAT.FLTTRIPD bit indicates whether the PWM Channel D fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.              |
|         |          | 0 Channel D Fault Trip Status is "not tripped"                                                                                                                                                           |
|         |          | 1 Channel D Fault Trip Status is "tripped"                                                                                                                                                               |
| 9       | SRTRIPC  | Self-Restart Trip Status for Channel C.<br>The PWM_STAT.SRTRIPC bit indicates whether the PWM Channel C self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0 Channel C Self-Restart Trip Status is "not tripped"                                                                                                                                                    |
|         |          | 1 Channel C Self-Restart Trip Status is "tripped"                                                                                                                                                        |
| 8       | FLTTRIPC | Fault Trip Status for Channel C.<br>The PWM_STAT.FLTTRIPC bit indicates whether the PWM Channel C fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.              |
|         |          | 0 Channel C Fault Trip Status is "not tripped"                                                                                                                                                           |
|         |          | 1 Channel C Fault Trip Status is "tripped"                                                                                                                                                               |
| 7       | SRTRIPB  | Self-Restart Trip Status for Channel B.<br>The PWM_STAT.SRTRIPB bit indicates whether the PWM Channel B self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0 Channel B Self-Restart Trip Status is "not tripped"                                                                                                                                                    |
|         |          | 1 Channel B Self-Restart Trip Status is "tripped"                                                                                                                                                        |

**Table 18-9:** PWM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | FLTTRIPB | Fault Trip Status for Channel B.<br>The PWM_STAT.FLTTRIPB bit indicates whether the PWM Channel B fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.              |
|         |          | 0 Channel B Fault Trip Status is "not tripped"                                                                                                                                                           |
|         |          | 1 Channel A Fault Trip Status is "tripped"                                                                                                                                                               |
| 5       | SRTRIPA  | Self-Restart Trip Status for Channel A.<br>The PWM_STAT.SRTRIPA bit indicates whether the PWM Channel A self-restart has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description. |
|         |          | 0 Channel A Self-Restart Trip Status is "not tripped"                                                                                                                                                    |
|         |          | 1 Channel A Self-Restart Trip Status is "tripped"                                                                                                                                                        |
| 4       | FLTTRIPA | Fault Trip Status for Channel A.<br>The PWM_STAT.FLTTRIPA bit indicates whether the PWM Channel A fault has been tripped. For more information, see the PWM_TRIPCFG.MODE0A bit description.              |
|         |          | 0 Channel A Fault Trip Status is "not tripped"                                                                                                                                                           |
|         |          | 1 Channel A Fault Trip Status is "tripped"                                                                                                                                                               |
| 3       | RAWTRIP1 | Raw Trip 1 Status.<br>The PWM_STAT.RAWTRIP1 bit indicates the raw input level for the PWM TRIP1 input.                                                                                                   |
|         |          | 0 TRIP1 Level is Low                                                                                                                                                                                     |
|         |          | 1 TRIP1 Level is High                                                                                                                                                                                    |

**Table 18-9:** PWM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | RAWTRIP0 | Raw Trip 0 Status.<br>The <code>PWM_STAT.RAWTRIP0</code> bit indicates the raw input level for the PWM TRIP0 input.                                              |
|         |          | 0      TRIP0 Level is Low                                                                                                                                        |
|         |          | 1      TRIP0 Level is High                                                                                                                                       |
| 1       | TRIP1    | Status bit set when TRIP1 is active low.<br>The <code>PWM_STAT.TRIP1</code> bit indicates whether the PWM TRIP1 fault has been tripped with an active-low input. |
|         |          | 0      TRIP1 status is "not tripped"                                                                                                                             |
|         |          | 1      TRIP1 status is "tripped" (active low)                                                                                                                    |
| 0       | TRIP0    | Status bit set when TRIP0 is active low.<br>The <code>PWM_STAT.TRIP0</code> bit indicates whether the PWM TRIP0 fault has been tripped with an active-low input. |
|         |          | 0      TRIP0 status is "not tripped"                                                                                                                             |
|         |          | 1      TRIP0 status is "tripped" (active low)                                                                                                                    |

## Interrupt Mask Register

The `PWM_IMSK` register masks (disables) or unmask (enables) PWM interrupts. When an unmasked interrupt occurs, the PWM latches the interrupt status in the `PWM_ILAT` register.

PWM\_IMSK: Interrupt Mask Register - R/W

Reset = 0x0000 0000

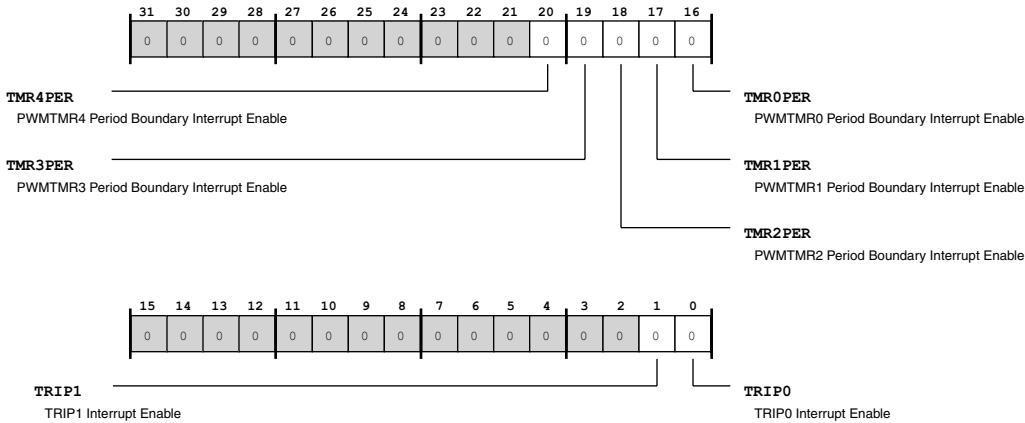


Figure 18-25: PWM\_IMSK Register Diagram

Table 18-10: PWM\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | TMR4PER  | PWMTMR4 Period Boundary Interrupt Enable.<br>The PWM_IMSK.TMR4PER bit enables (unmasks) the PWMTMR4 period boundary interrupt. This condition occurs when the timer's period boundary is reached (PWM_STAT.TMR4PER =1). |
|         |          | 0 Mask PWMTMR4 Period Interrupt                                                                                                                                                                                         |
|         |          | 1 Unmask PWMTMR4 Period Interrupt                                                                                                                                                                                       |
| 19      | TMR3PER  | PWMTMR3 Period Boundary Interrupt Enable.<br>The PWM_IMSK.TMR3PER bit enables (unmasks) the PWMTMR3 period boundary interrupt. This condition occurs when the timer's period boundary is reached (PWM_STAT.TMR3PER =1). |
|         |          | 0 Mask PWMTMR3 Period Interrupt                                                                                                                                                                                         |
|         |          | 1 Unmask PWMTMR3 Period Interrupt                                                                                                                                                                                       |

**Table 18-10:** PWM\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | TMR2PER  | PWMTMR2 Period Boundary Interrupt Enable.<br>The PWM_IMSK.TMR2PER bit enables (unmasks) the PWMTMR2 period boundary interrupt. This condition occurs when the timer's period boundary is reached (PWM_STAT.TMR2PER =1). |
|         |          | 0 Mask PWMTMR2 Period Interrupt                                                                                                                                                                                         |
|         |          | 1 Unmask PWMTMR2 Period Interrupt                                                                                                                                                                                       |
| 17      | TMR1PER  | PWMTMR1 Period Boundary Interrupt Enable.<br>The PWM_IMSK.TMR1PER bit enables (unmasks) the PWMTMR1 period boundary interrupt. This condition occurs when the timer's period boundary is reached (PWM_STAT.TMR1PER =1). |
|         |          | 0 Mask PWMTMR1 Period Interrupt                                                                                                                                                                                         |
|         |          | 1 Unmask PWMTMR1 Period Interrupt                                                                                                                                                                                       |
| 16      | TMR0PER  | PWMTMR0 Period Boundary Interrupt Enable.<br>The PWM_IMSK.TMR0PER bit enables (unmasks) the PWMTMR0 period boundary interrupt. This condition occurs when the timer's period boundary is reached (PWM_STAT.TMR0PER =1). |
|         |          | 0 Mask PWMTMR0 Period Interrupt                                                                                                                                                                                         |
|         |          | 1 Unmask PWMTMR0 Period Interrupt                                                                                                                                                                                       |
| 1       | TRIP1    | TRIP1 Interrupt Enable.<br>The PWM_IMSK.TRIP1 bit enables (unmasks) the TRIP1 interrupt. This condition occurs when fault input is tripped (PWM_STAT.TRIP1 =1).                                                         |
|         |          | 0 Mask TRIP1 Interrupt                                                                                                                                                                                                  |
|         |          | 1 Unmask TRIP1 Interrupt                                                                                                                                                                                                |

Table 18-10: PWM\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | TRIP0    | TRIP0 Interrupt Enable.<br>The PWM_IMSK.TRIP0 bit enables (unmasks) the TRIP0 interrupt. This condition occurs when fault input is tripped (PWM_STAT.TRIP0 =1). |
|         |          | 0 Mask TRIP0 Interrupt                                                                                                                                          |
|         |          | 1 Unmask TRIP0 Interrupt                                                                                                                                        |

Interrupt Latch Register

The PWM\_ILAT register latches the occurrence of unmasked (enabled) PWM interrupts. These interrupts are unmasked or masked with the PWM\_IMSK register.

PWM\_ILAT: Interrupt Latch Register - R/W

Reset = 0x0000 0000

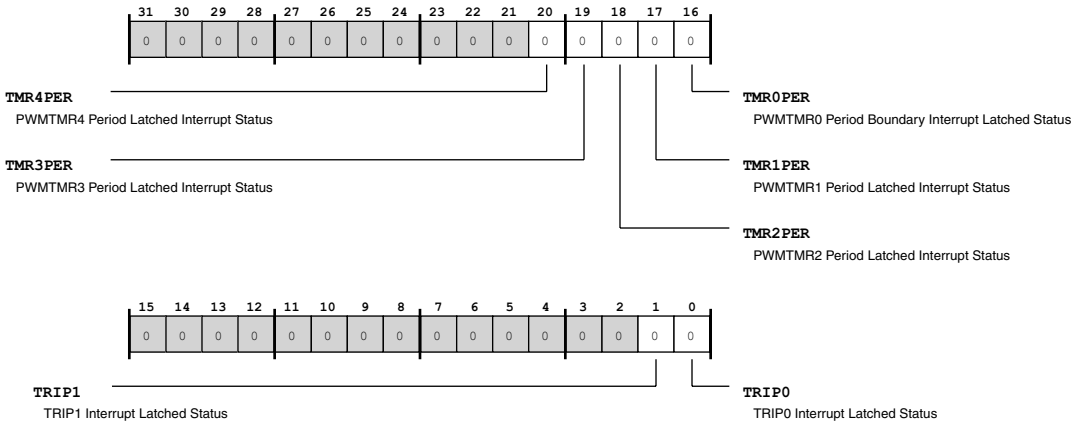


Figure 18-26: PWM\_ILAT Register Diagram

**Table 18-11:** PWM\_ILAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | TMR4PER  | PWMTMR4 Period Latched Interrupt Status.<br>The PWM_ILAT.TMR4PER bit indicates the latched status of the PWMTMR4 period boundary interrupt.          |
|         |          | 0 No Interrupt Latched                                                                                                                               |
|         |          | 1 Interrupt Latched                                                                                                                                  |
| 19      | TMR3PER  | PWMTMR3 Period Latched Interrupt Status.<br>The PWM_ILAT.TMR3PER bit indicates the latched status of the PWMTMR3 period boundary interrupt.          |
|         |          | 0 No Interrupt Latched                                                                                                                               |
|         |          | 1 Interrupt Latched                                                                                                                                  |
| 18      | TMR2PER  | PWMTMR2 Period Latched Interrupt Status.<br>The PWM_ILAT.TMR2PER bit indicates the latched status of the PWMTMR2 period boundary interrupt.          |
|         |          | 0 No Interrupt Latched                                                                                                                               |
|         |          | 1 Interrupt Latched                                                                                                                                  |
| 17      | TMR1PER  | PWMTMR1 Period Latched Interrupt Status.<br>The PWM_ILAT.TMR1PER bit indicates the latched status of the PWMTMR1 period boundary interrupt.          |
|         |          | 0 No Interrupt Latched                                                                                                                               |
|         |          | 1 Interrupt Latched                                                                                                                                  |
| 16      | TMR0PER  | PWMTMR0 Period Boundary Interrupt Latched Status.<br>The PWM_ILAT.TMR0PER bit indicates the latched status of the PWMTMR0 period boundary interrupt. |
|         |          | 0 No Interrupt Latched                                                                                                                               |
|         |          | 1 Interrupt Latched                                                                                                                                  |

Table 18-11: PWM\_ILAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                        |                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------|----------------------|
| 1       | TRIP1    | TRIP1 Interrupt Latched Status.<br>The PWM_ILAT.TRIP1 bit indicates the latched status of the TRIP1 interrupt. |                      |
|         |          | 0                                                                                                              | No Interrupt Latched |
|         |          | 1                                                                                                              | Interrupt Latched    |
| 0       | TRIP0    | TRIP0 Interrupt Latched Status.<br>The PWM_ILAT.TRIP0 bit indicates the latched status of the TRIP0 interrupt. |                      |
|         |          | 0                                                                                                              | No Interrupt Latched |
|         |          | 1                                                                                                              | Interrupt Latched    |

Chop Configuration Register

The PWM\_CHOPCFG register holds a divisor value that controls the chopping frequency. The PWM permits a mixing of the output signals with a high-frequency chopping signal to aid with interfacing to pulse transformers. Also note that high-frequency chopping may be independently enabled for each channel's high-side and the low-side outputs using channel control bits. (For example, control chopping for Channel A with the PWM\_CHANCFG.ENCHOPAH and PWM\_CHANCFG.ENCHOPAL bits.)

PWM\_CHOPCFG: Chop Configuration Register - R/W

Reset = 0x0000 0000

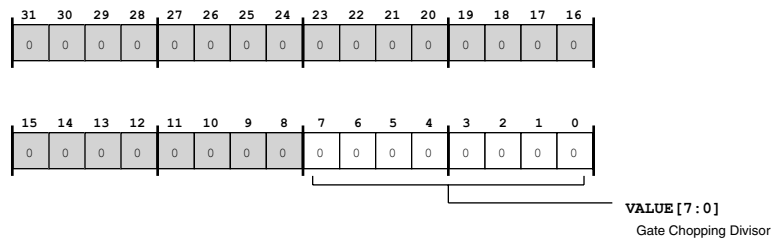


Figure 18-27: PWM\_CHOPCFG Register Diagram



**Table 18-12:** PWM\_CHOPCFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | <p>Gate Chopping Divisor.</p> <p>The PWM_CHOPCFG.VALUE bits provide the high frequency chopping divisor. When the divisor value is changed, the new period takes effect from the next edge of the chopping signal. The PWM_CHOPCFG.VALUE value may be calculated using either of the following formulas:</p> $\text{CHOPDIV} = [(T_{\text{CHOP}}/T_{\text{CK}}) / 4] - 1$ $\text{CHOPDIV} = [(f_{\text{CK}} / f_{\text{CHOP}}) / 4] - 1$ |

## Dead Time Register

The PWM\_DT register controls the dead time, which the PWM inserts into the pairs of output signals. Note that each channel has its own version of a double buffered dead time register, the double buffering of which depends on the period boundary of the PWMTMRx that the channel could be currently using. The dead time,  $T_d$ , is related to the value in the PWM\_DT register by:

$$T_d = \text{PWM\_DT} \times 2 \times t_{\text{CK}}$$

Note that the PWM holds the buffered PWM\_DT value for a channel at 0 if the channel's low side mode is independent (for example, PWM\_CHANCFG.MODELSA = 1). Also, note that the PWM\_DT value must be less than half the respective timer period (for example, PWM\_TMO/2). For more information about applying dead time to PWM output pairs, see the PWM Functional Description section.

### PWM\_DT: Dead Time Register - R/W

Reset = 0x0000 0000

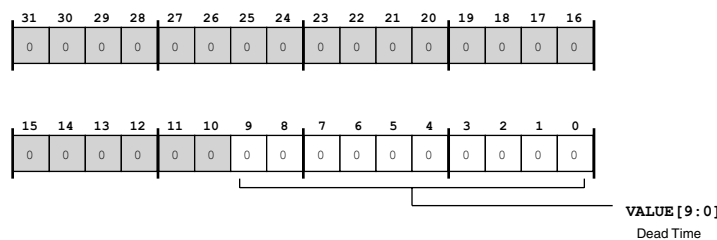

**Figure 18-28:** PWM\_DT Register Diagram

Table 18-13: PWM\_DT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------|
| 9:0     | VALUE    | Dead Time.<br>The PWM_DT.VALUE bits select the dead time that the PWM adds to the timing of the output pairs. |

Sync Pulse Width Register

The PWM\_SYNC\_WID register selects the pulse width for the external sync pulse available on the PWM\_SYNC pin. The relation between the PWM\_SYNC\_WID register value and the pulse width ( $T_{PWM\_SYNC}$ ) is give by the formula:

$$PWM\_SYNC\_WID = (T_{PWM\_SYNC} / t_{CK}) - 1$$

For more information about applying the sync pulse width, see the PWM Functional Description section. Note that if the pulse width is changed in between sync pulses, the PWM applies the changed width on the next internal sync pulse. If, while the sync pulse is active, the chosen timer reaches its period boundary, the changed pulse width takes effect on that period boundary.

PWM\_SYNC\_WID: Sync Pulse Width Register - R/W

Reset = 0x0000 03ff

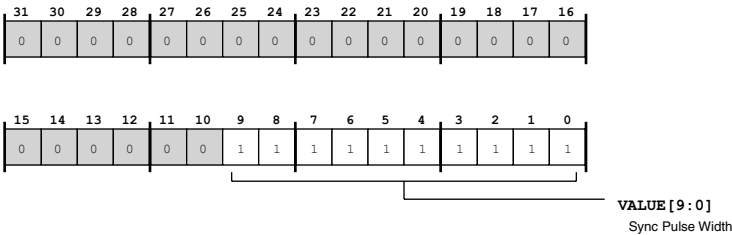


Figure 18-29: PWM\_SYNC\_WID Register Diagram

Table 18-14: PWM\_SYNC\_WID Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 9:0     | VALUE    | Sync Pulse Width.<br>The PWM_SYNC_WID.VALUE bits select the pulse width for the external sync pulse available on the PWM_SYNC pin. |

## Timer 0 Period Register

The PWM\_TM0 register controls the switch period ( $T_{SP}$  of the PWMTMR0 timer. The PWM\_TM0 value is in units of  $t_{CK}$  (the period of the peripheral clock) and the and is given by the formula:

$$PWM\_TM0 = (T_{SP}) / 2 \times t_{CK}$$

The value written to the register is effectively the number of  $t_{CK}$  clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM\_TM0 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

**PWM\_TM0: Timer 0 Period Register - R/W**

Reset = 0x0000 0000

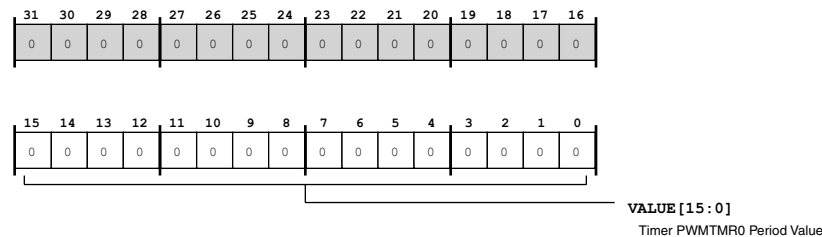


**Figure 18-30: PWM\_TM0 Register Diagram**

**Table 18-15: PWM\_TM0 Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Timer PWMTMR0 Period Value.<br>The PWM_TM0.VALUE bits select the period for the PWMTMR0 timer. |

## Timer 1 Period Register

The PWM\_TM1 register controls the switch period ( $T_{SP}$  of the PWMTMR1 timer. The PWM\_TM1 value is in units of  $t_{CK}$  (the period of the peripheral clock) and the and is given by the formula:

$$PWM\_TM1 = (T_{SP}) / 2 \times t_{CK}$$

The value written to the register is effectively the number of  $t_{CK}$  clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM\_TM1 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

PWM\_TM1: Timer 1 Period Register - R/W

Reset = 0x0000 0000

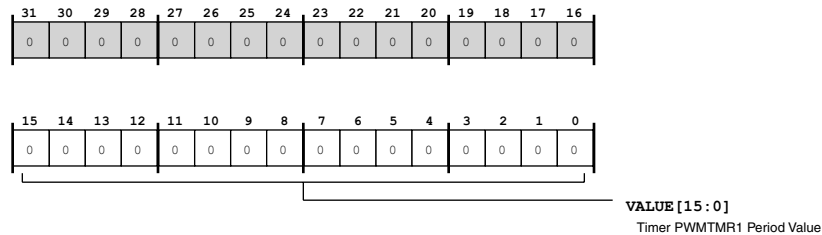


Figure 18-31: PWM\_TM1 Register Diagram

Table 18-16: PWM\_TM1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Timer PWMTMR1 Period Value.<br>The PWM_TM1 . VALUE bits select the period for the PWMTMR1 timer. |

Timer 2 Period Register

The PWM\_TM2 register controls the switch period ( $T_{SP}$  of the PWMTMR2 timer. The PWM\_TM2 value is in units of  $t_{CK}$  (the period of the peripheral clock) and the and is given by the formula:

$$PWM\_TM1 = (T_{SP}) / 2 \times t_{CK}$$

The value written to the register is effectively the number of  $t_{CK}$  clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM\_TM2 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

**PWM\_TM2: Timer 2 Period Register - R/W**

Reset = 0x0000 0000

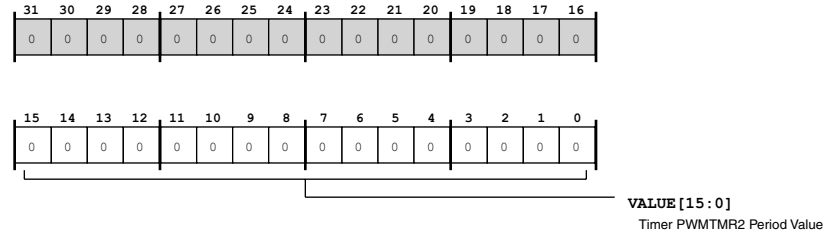


**Figure 18-32:** PWM\_TM2 Register Diagram

**Table 18-17:** PWM\_TM2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Timer PWMTMR2 Period Value.<br>The PWM_TM2.VALUE bits select the period for the PWMTMR2 timer. |

## Timer 3 Period Register

The PWM\_TM3 register controls the switch period ( $T_{SP}$  of the PWMTMR3 timer. The PWM\_TM3 value is in units of  $t_{CK}$  (the period of the peripheral clock) and the and is given by the formula:

$$PWM\_TM3 = (T_{SP}) / 2 \times t_{CK}$$

The value written to the register is effectively the number of  $t_{CK}$  clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM\_TM3 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

PWM\_TM3: Timer 3 Period Register - R/W

Reset = 0x0000 0000

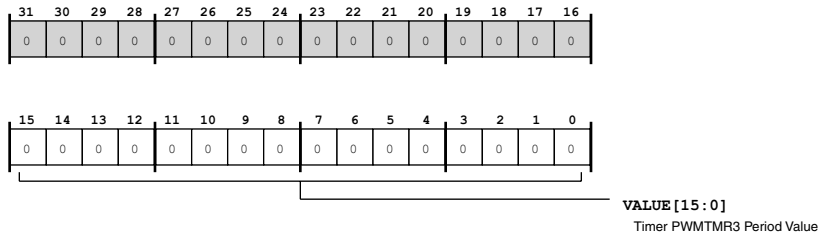


Figure 18-33: PWM\_TM3 Register Diagram

Table 18-18: PWM\_TM3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Timer PWMTMR3 Period Value.<br>The PWM_TM3.VALUE bits select the period for the PWMTMR3 timer. |

Timer 4 Period Register

The PWM\_TM4 register controls the switch period ( $T_{SP}$  of the PWMTMR4 timer. The PWM\_TM4 value is in units of  $t_{CK}$  (the period of the peripheral clock) and the and is given by the formula:

$$PWM\_TM4 = (T_{SP}) / 2 \times t_{CK}$$

The value written to the register is effectively the number of  $t_{CK}$  clock increments in half the period of the respective timer. For more information about applying the switch period, see the PWM Functional Description section. Note that PWM\_TM4 values of 0 and 1 are not defined and must not be used when the PWM is enabled.

**PWM\_TM4: Timer 4 Period Register - R/W**

Reset = 0x0000 0000

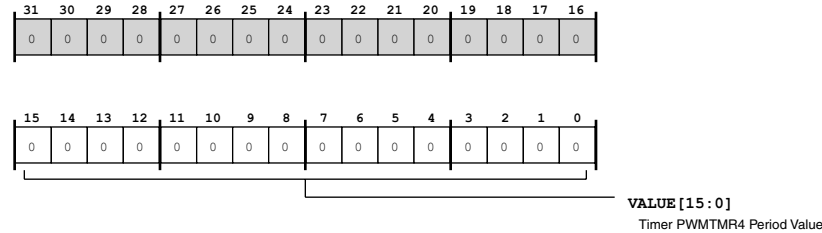


**Figure 18-34:** PWM\_TM4 Register Diagram

**Table 18-19:** PWM\_TM4 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Timer PWMTMR4 Period Value.<br>The PWM_TM4.VALUE bits select the period for the PWMTMR4 timer. |

## Channel A Delay Register

The PWM\_DLYA register controls a delay for the Channel A timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM\_CTL.DLYAEN =1}. For more information about applying the delay, see the PWM Functional Description section. Note that the PWM\_DLYA delay value must be less than twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM\_DLYA must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

**PWM\_DLYA: Channel A Delay Register - R/W**

Reset = 0x0000 0000

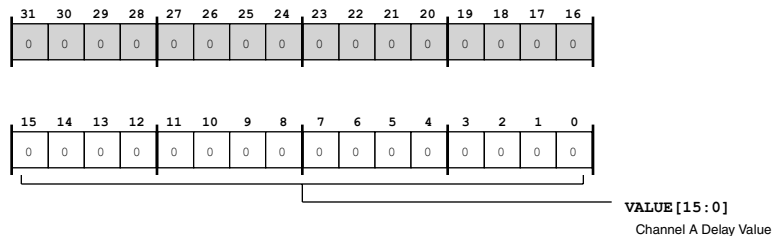


**Figure 18-35:** PWM\_DLYA Register Diagram

Table 18-20: PWM\_DLYA Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Channel A Delay Value.<br>The PWM_DLYA.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel A. |

Channel B Delay Register

The PWM\_DLYB register controls a delay for the Channel B timer (only PWMTMR1, PWMTMR2, PWMTMR3 or PWMTMR4) with reference to the main timer (PWMTMR0). To use apply this delay, the delay must be enabled (PWM\_CTL.DLYBEN =1}. For more information about applying the delay, see the PWM Functional Description section. Note that the PWM\_DLYB delay value must be less that less that twice the period value of the timer being used for the channel (for example, if PWMTMR1 is used, PWM\_DLYB must be less than 2xPWMTMR1). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if PWMTMR1 is used, PWMTMR0 = NxPWMTMR1, where N is an integer).

PWM\_DLYB: Channel B Delay Register - R/W

Reset = 0x0000 0000

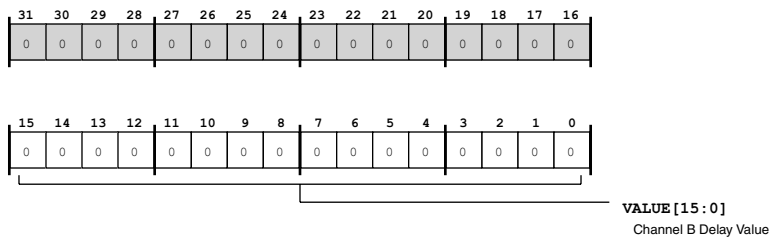


Figure 18-36: PWM\_DLYB Register Diagram

Table 18-21: PWM\_DLYB Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Channel B Delay Value.<br>The PWM_DLYB.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel B. |



## Channel C Delay Register

The `PWM_DLYC` register controls a delay for the Channel C timer (only `PWMTMR1`, `PWMTMR2`, `PWMTMR3` or `PWMTMR4`) with reference to the main timer (`PWMTMR0`). To use apply this delay, the delay must be enabled (`PWM_CTL.DLYCEN = 1`). For more information about applying the delay, see the PWM Functional Description section. Note that the `PWM_DLYC` delay value must be less than twice the period value of the timer being used for the channel (for example, if `PWMTMR1` is used, `PWM_DLYC` must be less than  $2 \times \text{PWMTMR1}$ ). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if `PWMTMR1` is used,  $\text{PWMTMR0} = N \times \text{PWMTMR1}$ , where  $N$  is an integer).

**PWM\_DLYC: Channel C Delay Register - R/W**

Reset = 0x0000 0000

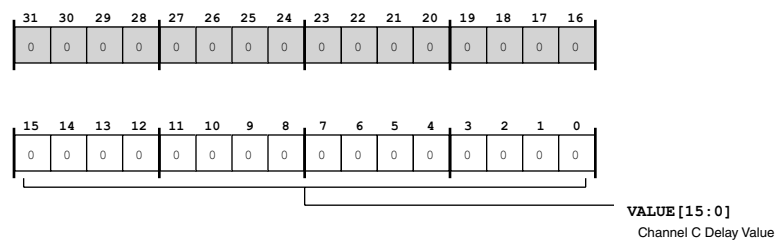


**Figure 18-37:** PWM\_DLYC Register Diagram

**Table 18-22:** PWM\_DLYC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Channel C Delay Value.<br>The <code>PWM_DLYC.VALUE</code> bits select the phase delay between the main timer ( <code>PWMTMR0</code> ) and the timer used for Channel C. |

## Channel D Delay Register

The `PWM_DLYD` register controls a delay for the Channel D timer (only `PWMTMR1`, `PWMTMR2`, `PWMTMR3` or `PWMTMR4`) with reference to the main timer (`PWMTMR0`). To use apply this delay, the delay must be enabled (`PWM_CTL.DLYDEN = 1`). For more information about applying the delay, see the PWM Functional Description section. Note that the `PWM_DLYD` delay value must be less than twice the period value of the timer being used for the channel (for example, if `PWMTMR1` is used, `PWM_DLYD` must be less than  $2 \times \text{PWMTMR1}$ ). Also, note that the period of the main timer must be an integer multiple of the timer being used for the channel (for example, if `PWMTMR1` is used,  $\text{PWMTMR0} = N \times \text{PWMTMR1}$ , where  $N$  is an integer).

PWM\_DLYD: Channel D Delay Register - R/W

Reset = 0x0000 0000

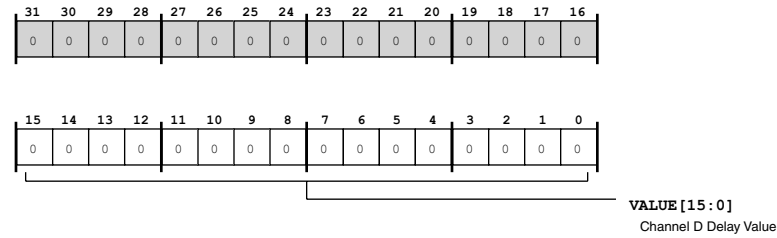


Figure 18-38: PWM\_DLYD Register Diagram

Table 18-23: PWM\_DLYD Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Channel D Delay Value.<br>The PWM_DLYD.VALUE bits select the phase delay between the main timer (PWMTMR0) and the timer used for Channel D. |

Channel A Control Register

The PWM\_ACTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

PWM\_ACTL: Channel A Control Register - R/W

Reset = 0x0000 0000

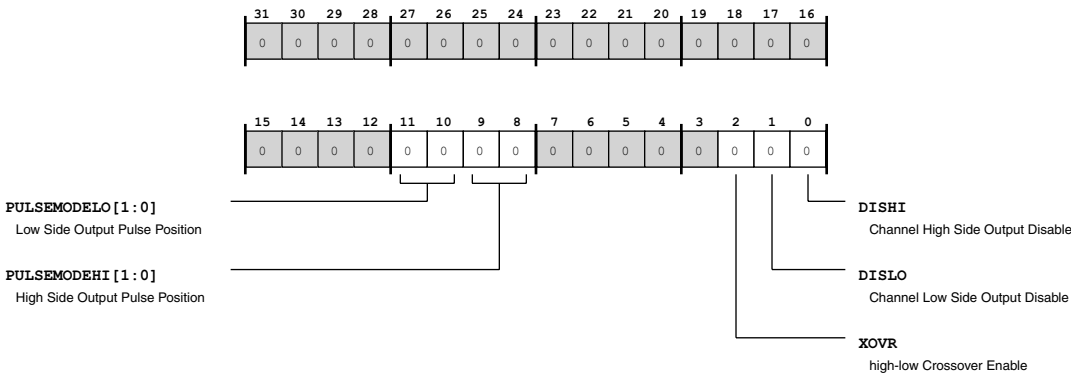


Figure 18-39: PWM\_ACTL Register Diagram

**Table 18-24:** PWM\_ACTL Register Fields

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:10   | PULSEMODELO | <p>Low Side Output Pulse Position.</p> <p>The PWM_ACTL.PULSEMODELO bits select the pulse position for Channel A low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_AL0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_AL0 and PWM_AL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_AL0 and PWM_AL1).</p> |
|         |             | 0 Symmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |             | 1 Asymmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |             | 2 Left Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |             | 3 Right Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Table 18-24:** PWM\_ACTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                         |
|---------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 9:8     | PULSEMODEHI | High Side Output Pulse Position.<br>The PWM_ACTL.PULSEMODEHI bits select the pulse position for Channel A high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_AH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_AH0 and PWM_AH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_AH0 and PWM_AH1). |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Symmetrical             |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Asymmetrical            |
|         |             | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Left Half               |
|         |             | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Right Half              |
| 2       | XOVR        | high-low Crossover Enable.<br>The PWM_ACTL.XOVR bit enables crossover between the channel's high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Disable Crossover       |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Enable Crossover        |
| 1       | DISLO       | Channel Low Side Output Disable.<br>The PWM_ACTL.DISLO bit enables the channel's low side output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Disable Low Side Output |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Enable Low Side Output  |

**Table 18-24:** PWM\_ACTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------|
| 0       | DISHI    | Channel High Side Output Disable.<br>The PWM_ACTL.DISHI bit enables the channel's high side output. |
|         |          | 0      Disable High Side Output                                                                     |
|         |          | 1      Enable High Side Output                                                                      |

## Channel A-High Duty-0 Register

The PWM\_AH0 and PWM\_AH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the high side output pulses for the Channel A duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_ACTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_AH0 register to determine the assertion and de-assertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel A high pulse output for count less than PWM\_AH0 and de-asserts this output for count greater than PWM\_AH1.

The value range for the PWM\_AH0 and PWM\_AH1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_AH0 and PWM\_AH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_AH0 or PWM\_AH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

PWM\_AH0: Channel A-High Duty-0 Register - R/W

Reset = 0x0000 0000

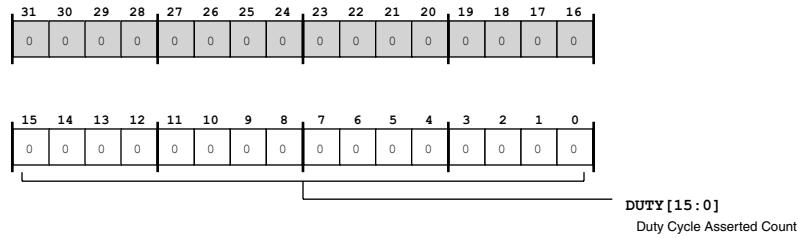


Figure 18-40: PWM\_AH0 Register Diagram

Table 18-25: PWM\_AH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_AH0.DUTY bits select the duty cycle asserted count for Channel A high side output. |

Channel A-High Duty-1 Register

The PWM\_AH0 and PWM\_AH1 registers determine the width for the high side output pulses. For more information, see the PWM\_AH0 register description.

PWM\_AH1: Channel A-High Duty-1 Register - R/W

Reset = 0x0000 0000

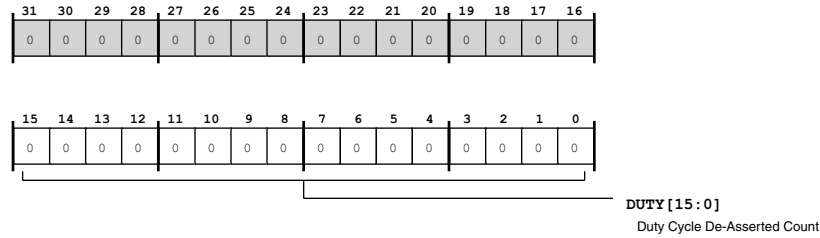


Figure 18-41: PWM\_AH1 Register Diagram

**Table 18-26:** PWM\_AH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_AH1.DUTY bits select the duty cycle de-asserted count for Channel A high side output. |

## Channel A-Low Duty-0 Register

The PWM\_AL0 and PWM\_AL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the low side output pulses for the Channel A duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_ACTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_AL0 register to determine the assertion and de-assertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel A low pulse output for count less than PWM\_AL0 and de-asserts this output for count greater than PWM\_AL1.

The value range for the PWM\_AL0 and PWM\_AL1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_AL0 and PWM\_AL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_AL0 or PWM\_AL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

PWM\_AL0: Channel A-Low Duty-0 Register - R/W

Reset = 0x0000 0000

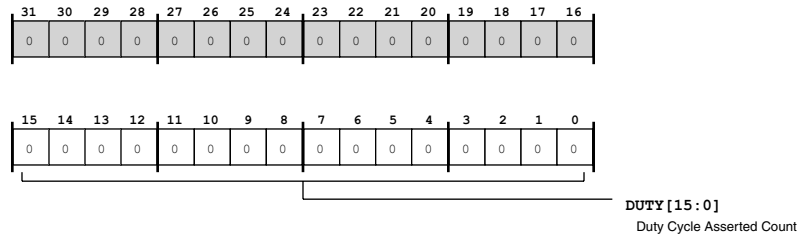


Figure 18-42: PWM\_AL0 Register Diagram

Table 18-27: PWM\_AL0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_AL0.DUTY bits select the duty cycle asserted count for Channel A low side output. |

Channel A-Low Duty-1 Register

The PWM\_AL0 and PWM\_AL1 registers determine the width for the low side output pulses. For more information, see the PWM\_AL0 register description.

PWM\_AL1: Channel A-Low Duty-1 Register - R/W

Reset = 0x0000 0000

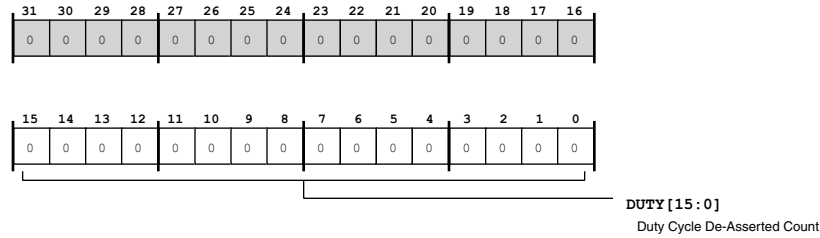


Figure 18-43: PWM\_AL1 Register Diagram



**Table 18-28:** PWM\_AL1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_AL1.DUTY bits select the duty cycle de-asserted count for Channel A low side output. |

## Channel B Control Register

The PWM\_BCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

**PWM\_BCTL: Channel B Control Register - R/W**

Reset = 0x0000 0000

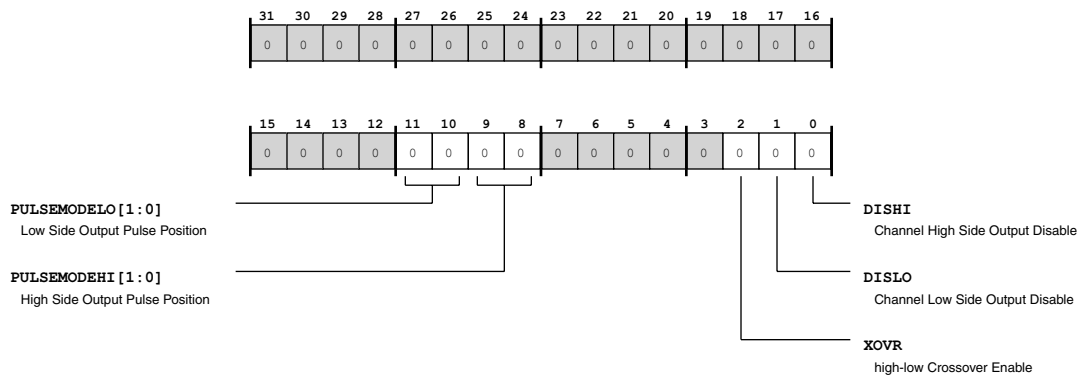


**Figure 18-44:** PWM\_BCTL Register Diagram

**Table 18-29:** PWM\_BCTL Register Fields

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |              |
|---------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| 11:10   | PULSEMODELO | Low Side Output Pulse Position.<br>The PWM_BCTL.PULSEMODELO bits select the pulse position for Channel B low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_BLO register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_BLO and PWM_BL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_BLO and PWM_BL1). |              |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Symmetrical  |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Asymmetrical |
|         |             | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Left Half    |
|         |             | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Right Half   |

**Table 18-29:** PWM\_BCTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                         |
|---------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 9:8     | PULSEMODEHI | High Side Output Pulse Position.<br>The PWM_BCTL.PULSEMODEHI bits select the pulse position for Channel B high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_BH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_BH0 and PWM_BH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_BH0 and PWM_BH1). |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Symmetrical             |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Asymmetrical            |
|         |             | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Left Half               |
|         |             | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Right Half              |
| 2       | XOVR        | high-low Crossover Enable.<br>The PWM_BCTL.XOVR bit enables crossover between the channel's high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Disable Crossover       |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Enable Crossover        |
| 1       | DISLO       | Channel Low Side Output Disable.<br>The PWM_BCTL.DISLO bit enables the channel's low side output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                         |
|         |             | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Disable Low Side Output |
|         |             | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Enable Low Side Output  |

**Table 18-29:** PWM\_BCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------|
| 0       | DISHI    | Channel High Side Output Disable.<br>The PWM_BCTL.DISHI bit enables the channel's high side output. |
|         |          | 0      Disable High Side Output                                                                     |
|         |          | 1      Enable High Side Output                                                                      |

## Channel B-High Duty-0 Register

The PWM\_BH0 and PWM\_BH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the high side output pulses for the Channel B duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_BCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_BH0 register to determine the assertion and de-assertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel B high pulse output for count less than PWM\_BH0 and de-asserts this output for count greater than PWM\_BH1.

The value range for the PWM\_BH0 and PWM\_BH1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_BH0 and PWM\_BH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_BH0 or PWM\_BH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

**PWM\_BH0: Channel B-High Duty-0 Register - R/W**

Reset = 0x0000 0000

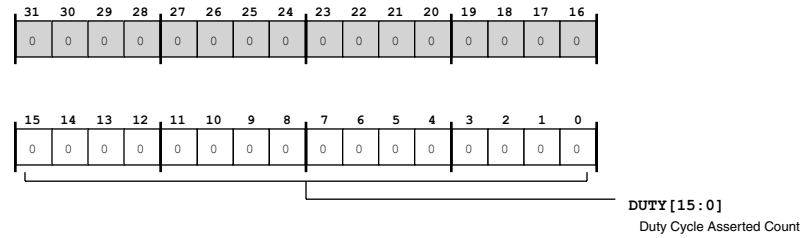


**Figure 18-45:** PWM\_BH0 Register Diagram

**Table 18-30:** PWM\_BH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count. |

## Channel B-High Duty-1 Register

The PWM\_BH0 and PWM\_BH1 registers determine the width for the high side output pulses. For more information, see the PWM\_BH0 register description.

**PWM\_BH1: Channel B-High Duty-1 Register - R/W**

Reset = 0x0000 0000

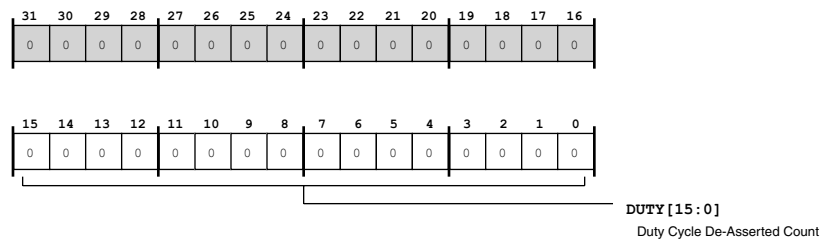


**Figure 18-46:** PWM\_BH1 Register Diagram

**Table 18-31:** PWM\_BH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count. |

## Channel B-Low Duty-0 Register

The PWM\_BLO and PWM\_BL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the low side output pulses for the Channel B duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_BCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_BLO register to determine the assertion and de-assertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel B low pulse output for count less than PWM\_BLO and de-asserts this output for count greater than PWM\_BL1.

The value range for the PWM\_BLO and PWM\_BL1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_BLO and PWM\_BL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_BLO or PWM\_BL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

### PWM\_BLO: Channel B-Low Duty-0 Register - R/W

Reset = 0x0000 0000

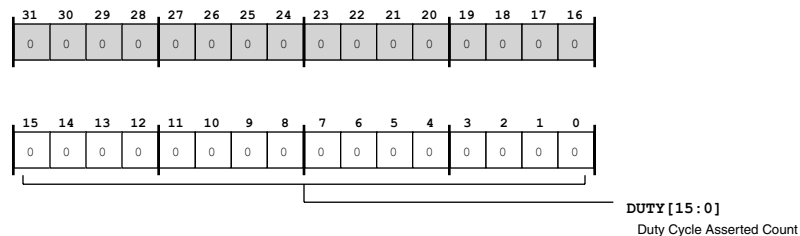


**Figure 18-47:** PWM\_BLO Register Diagram

**Table 18-32:** PWM\_BLO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_BLO.DUTY bits select the duty cycle asserted count for Channel B low side output. |

## Channel B-Low Duty-1 Register

The PWM\_BLO and PWM\_BL1 registers determine the width for the low side output pulses. For more information, see the PWM\_BLO register description.

**PWM\_BL1: Channel B-Low Duty-1 Register - R/W**

Reset = 0x0000 0000

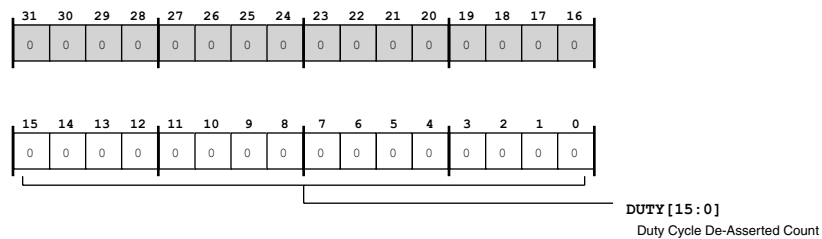


**Figure 18-48:** PWM\_BL1 Register Diagram

**Table 18-33:** PWM\_BL1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_BL1.DUTY bits select the duty cycle de-asserted count for Channel B low side output. |

## Channel C Control Register

The PWM\_CCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

PWM\_CCTL: Channel C Control Register - R/W

Reset = 0x0000 0000

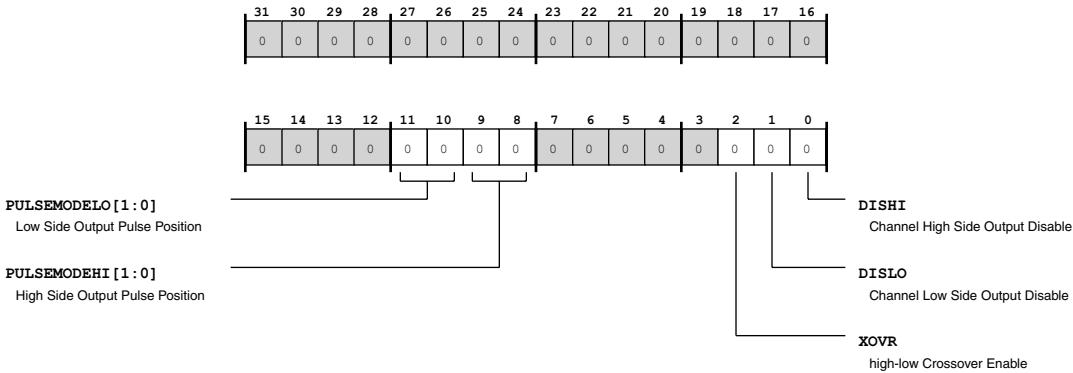


Figure 18-49: PWM\_CCTL Register Diagram

Table 18-34: PWM\_CCTL Register Fields

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:10   | PULSEMODELO | Low Side Output Pulse Position.<br>The PWM_CCTL.PULSEMODELO bits select the pulse position for Channel C low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_CLO register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_CLO and PWM_CL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_CLO and PWM_CL1). |
|         |             | 0 Symmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |             | 1 Asymmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |             | 2 Left Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |             | 3 Right Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |



**Table 18-34:** PWM\_CCTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | PULSEMODEHI | <p>High Side Output Pulse Position.</p> <p>The PWM_CCTL.PULSEMODEHI bits select the pulse position for Channel C high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_CH0 register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_CH0 and PWM_CH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_CH0 and PWM_CH1).</p> |
|         |             | 0 Symmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |             | 1 Asymmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |             | 2 Left Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |             | 3 Right Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 2       | XOVR        | <p>high-low Crossover Enable.</p> <p>The PWM_CCTL.XOVR bit enables crossover between the channel's high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |             | 0 Disable Crossover                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |             | 1 Enable Crossover                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 1       | DISLO       | <p>Channel Low Side Output Disable.</p> <p>The PWM_CCTL.DISLO bit enables the channel's low side output.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |             | 0 Disable Low Side Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |             | 1 Enable Low Side Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 18-34:** PWM\_CCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------|
| 0       | DISHI    | Channel High Side Output Disable.<br>The PWM_CCTL.DISHI bit enables the channel's high side output. |
|         |          | 0 Disable High Side Output                                                                          |
|         |          | 1 Enable High Side Output                                                                           |

## Channel C-High Pulse Duty Register 0

The PWM\_CH0 and PWM\_CH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the high side output pulses for the Channel C duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_CCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_CH0 register to determine the assertion and de-assertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel C high pulse output for count less than PWM\_CH0 and de-asserts this output for count greater than PWM\_CH1.

The value range for the PWM\_CH0 and PWM\_CH1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_CH0 and PWM\_CH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_CH0 or PWM\_CH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

PWM\_CH0: Channel C-High Pulse Duty Register 0 - R/W

Reset = 0x0000 0000

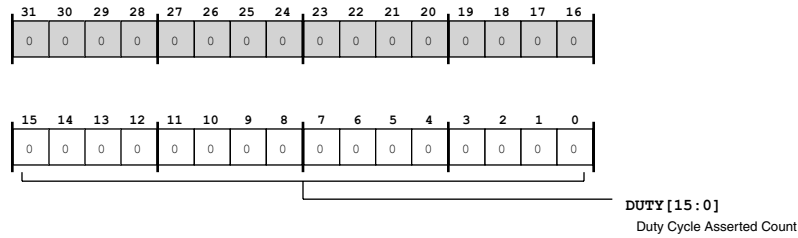


Figure 18-50: PWM\_CH0 Register Diagram

Table 18-35: PWM\_CH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_CH0.DUTY bits select the duty cycle asserted count for Channel C high side output. |

Channel C-High Pulse Duty Register 1

The PWM\_CH0 and PWM\_CH1 registers determine the width for the high side output pulses. For more information, see the PWM\_CH0 register description.

PWM\_CH1: Channel C-High Pulse Duty Register 1 - R/W

Reset = 0x0000 0000

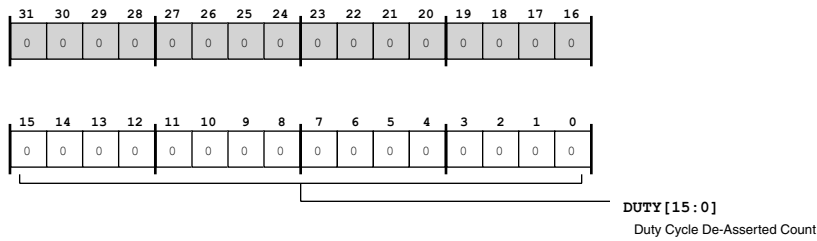


Figure 18-51: PWM\_CH1 Register Diagram

**Table 18-36:** PWM\_CH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_CH1.DUTY bits select the duty cycle de-asserted count for Channel C high side output. |

## Channel C-Low Pulse Duty Register 0

The PWM\_CL0 and PWM\_CL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the low side output pulses for the Channel C duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_CCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_CL0 register to determine the assertion and de-assertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel C low pulse output for count less than PWM\_CL0 and de-asserts this output for count greater than PWM\_CL1.

The value range for the PWM\_CL0 and PWM\_CL1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_CL0 and PWM\_CL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_CL0 or PWM\_CL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

**PWM\_CL0: Channel C-Low Pulse Duty Register 0 - R/W**

Reset = 0x0000 0000

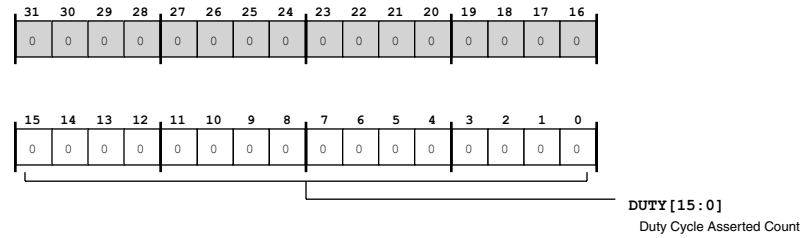


**Figure 18-52:** PWM\_CL0 Register Diagram

**Table 18-37:** PWM\_CL0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_CL0.DUTY bits select the duty cycle asserted count for Channel C low side output. |

## Channel C-Low Duty-1 Register

**PWM\_CL1: Channel C-Low Duty-1 Register - R/W**

Reset = 0x0000 0000

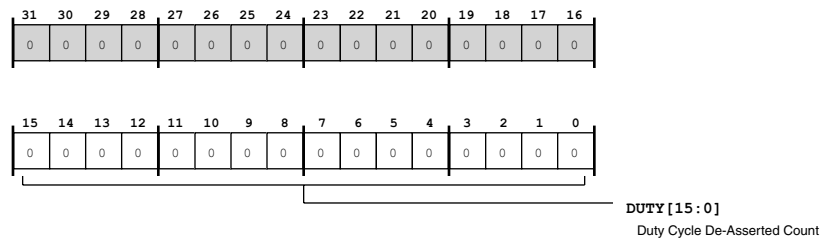


**Figure 18-53:** PWM\_CL1 Register Diagram

**Table 18-38:** PWM\_CL1 Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count. |

## Channel D Control Register

The PWM\_DCTL register selects the low and high side output pulse mode, enables low and high side output, and enables low/high side output crossover.

PWM\_DCTL: Channel D Control Register - R/W

Reset = 0x0000 0000

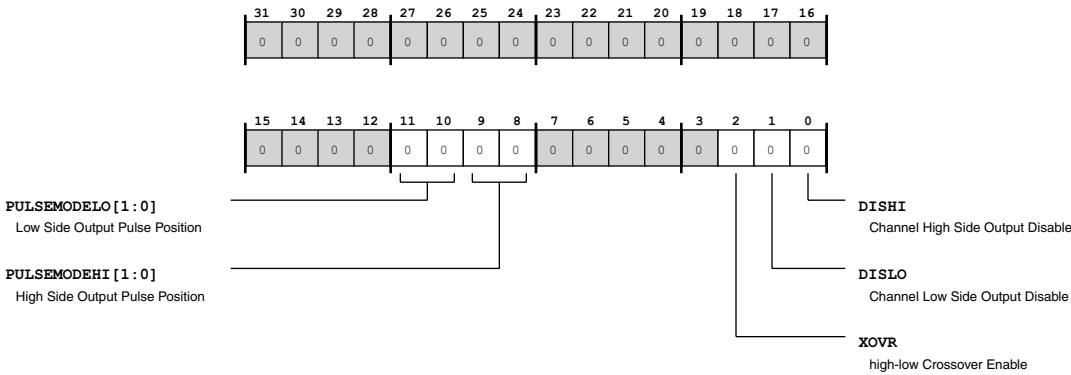


Figure 18-54: PWM\_DCTL Register Diagram

Table 18-39: PWM\_DCTL Register Fields

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:10   | PULSEMODELO | Low Side Output Pulse Position.<br>The PWM_DCTL.PULSEMODELO bits select the pulse position for Channel D low side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_DLO register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_DLO and PWM_DL1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_DLO and PWM_DL1). |
|         |             | 0 Symmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |             | 1 Asymmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |             | 2 Left Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |             | 3 Right Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

**Table 18-39:** PWM\_DCTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | PULSEMODEHI | High Side Output Pulse Position.<br>The PWM_DCTL.PULSEMODEHI bits select the pulse position for Channel D high side output. In symmetrical mode, the channel forms a symmetrical pulse waveform around the centre of the PWM period. Only one of the duty cycle registers is used for an output in symmetrical mode. Note that in this mode, the values in the PWM_DHO register is scaled, such that a value of 0 produces 50% duty. In asymmetrical mode, the channel forms an asymmetrical pulse waveform around the centre of the PWM period. This mode uses both the duty-cycle registers (PWM_DHO and PWM_DH1). In left half or right half mode, the channel forms the pulse waveforms on either the first half (left) or the second half (right) of the PWM period. This mode uses both the duty-cycle registers (PWM_DHO and PWM_DH1). |
|         |             | 0 Symmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |             | 1 Asymmetrical                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |             | 2 Left Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |             | 3 Right Half                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 2       | XOVR        | high-low Crossover Enable.<br>The PWM_DCTL.XOVR bit enables crossover between the channel's high and low side outputs. When enabled, this bit directs the PWM to send the low-side output through the high-side output pin and the high-side output through the low side output pin.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |             | 0 Disable Crossover                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |             | 1 Enable Crossover                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 1       | DISLO       | Channel Low Side Output Disable.<br>The PWM_DCTL.DISLO bit enables the channel's low side output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |             | 0 Disable Low Side Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |             | 1 Enable Low Side Output                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 18-39:** PWM\_DCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------|
| 0       | DISHI    | Channel High Side Output Disable.<br>The PWM_DCTL.DISHI bit enables the channel's high side output. |
|         |          | 0      Disable High Side Output                                                                     |
|         |          | 1      Enable High Side Output                                                                      |

## Channel D-High Duty-0 Register

The PWM\_DHO and PWM\_DH1 registers determine the width for the high side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the high side output pulses for the Channel D duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_DCTL.PULSEMODEHI bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_DHO register to determine the assertion and de-assertion count for the high side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel D high pulse output for count less than PWM\_DHO and de-asserts this output for count greater than PWM\_DH1.

The value range for the PWM\_DHO and PWM\_DH1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_DHO and PWM\_DH1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_DHO or PWM\_DH1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.



PWM\_DH0: Channel D-High Duty-0 Register - R/W

Reset = 0x0000 0000

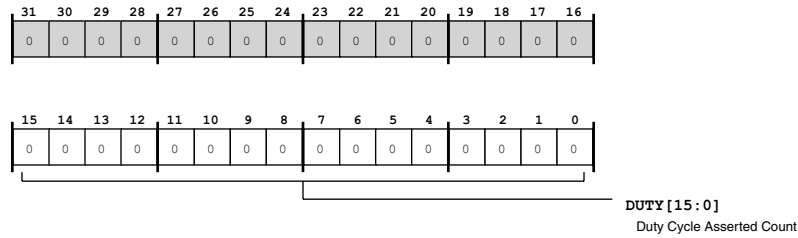


Figure 18-55: PWM\_DH0 Register Diagram

Table 18-40: PWM\_DH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_DH0.DUTY bits select the duty cycle asserted count for Channel D high side output. |

Channel D-High Pulse Duty Register 1

The PWM\_DH0 and PWM\_DH1 registers determine the width for the high side output pulses. For more information, see the PWM\_DH0 register description.

PWM\_DH1: Channel D-High Pulse Duty Register 1 - R/W

Reset = 0x0000 0000

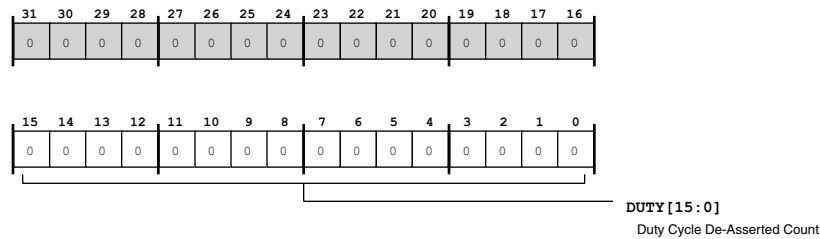


Figure 18-56: PWM\_DH1 Register Diagram

**Table 18-41:** PWM\_DH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_DH1.DUTY bits select the duty cycle de-asserted count for Channel D high side output. |

## Channel D-Low Pulse Duty Register 0

The PWM\_DL0 and PWM\_DL1 registers determine the width for the low side output pulses. The values in these registers select the assertion count (in terms of  $t_{CK}$ ) of the low side output pulses for the Channel D duty cycle.

The operation of the duty-cycle registers varies, depending on the pulse mode selected with the PWM\_DCTL.PULSEMODELO bits. When the pulse mode is symmetrical, the PWM uses the value in the PWM\_DL0 register to determine the assertion and de-assertion count for the low side output pulses. When the pulse mode is asymmetrical, left half, or right half, the PWM asserts Channel D low pulse output for count less than PWM\_DL0 and de-asserts this output for count greater than PWM\_DL1.

The value range for the PWM\_DL0 and PWM\_DL1 registers depends on the period of the timer being used by the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2$  (two's complement) and  $+PWM\_TMO/2$ , when dead time (PWM\_DT) is not considered.

When dead time is considered for symmetrical and asymmetrical pulse modes, the value range for PWM\_DL0 and PWM\_DL1 depends on the period of the time being used by the channel and the amount of dead time applied to the channel. For example, if PWM\_TMO is used, the duty cycle values may be between  $-PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $+PWM\_TMO/2 + PWM\_DT$ .

When dead time is considered for left half or right half pulse modes, if PWM\_TMO is used, the duty cycle values may be between  $PWM\_TMO/2 + PWM\_DT$  (two's complement) to  $-PWM\_TMO/2 - PWM\_DT$ .

Note that using values in the PWM\_DL0 or PWM\_DL1 registers that fall outside these limits causes PWM over or under modulation.

For more information about pulse modes and duty cycle selection, see the Functional Description section.

**PWM\_DL0: Channel D-Low Pulse Duty Register 0 - R/W**

Reset = 0x0000 0000

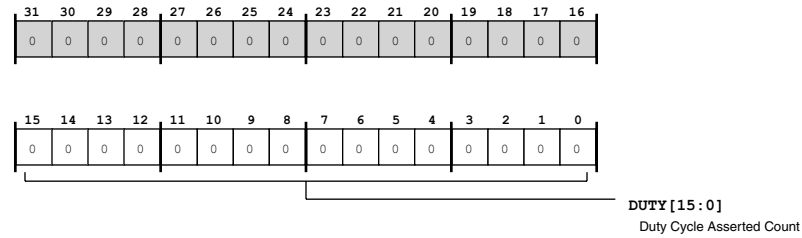


**Figure 18-57:** PWM\_DL0 Register Diagram

**Table 18-42:** PWM\_DL0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle Asserted Count.<br>The PWM_DL0.DUTY bits select the duty cycle asserted count for Channel D low side output. |

## Channel D-Low Pulse Duty Register 1

The PWM\_DL0 and PWM\_DL1 registers determine the width for the low side output pulses. For more information, see the PWM\_DL0 register description.

**PWM\_DL1: Channel D-Low Pulse Duty Register 1 - R/W**

Reset = 0x0000 0000

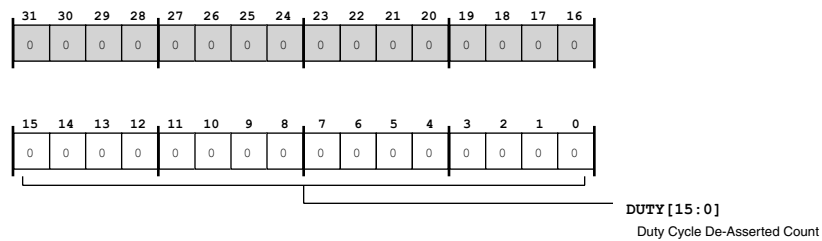


**Figure 18-58:** PWM\_DL1 Register Diagram

**Table 18-43:** PWM\_DL1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | DUTY     | Duty Cycle De-Asserted Count.<br>The PWM_DL1.DUTY bits select the duty cycle de-asserted count for Channel D low side output. |

# 19 Universal Async Rx/Tx (UART)

This chapter describes the Universal Asynchronous Receiver/Transmitter (UART) modules.

The UART modules are full-duplex peripherals compatible with PC-style industry-standard UARTs. The UARTs convert data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word length, stop bits, bit rate, and parity generation options. The UARTs include interrupt-handling hardware. Interrupts can be generated from multiple events.

The UARTs are logically compliant to EIA-232E, EIA-422, EIA-485 and LIN standards, but usually require external transceiver devices to meet electrical requirements. In IrDA (Infrared Data Association) mode, the UARTs meet the half-duplex IrDA SIR (9.6/115.2 Kbps rate) protocol. In multi-drop bus mode, the UARTs meet the full-duplex MDB/ICP v2.0 protocol.

Partial modem status and control functionality is supported by the UART module to allow for hardware flow control.

The UARTs are DMA-capable peripherals with support for separate transmit and receive DMA master channels. They can be used in either DMA or programmed core mode of operation. The core mode requires software management of the data flow using either interrupts or polling. The DMA method requires minimal software intervention as the DMA engine itself moves the data. Each UART has its own separate transmit and receive DMA channels.

One of the peripheral timers can be used to provide a hardware-assisted auto-baud detection mechanism for use with the UART. The timers are external to the UART.

## UART Features

Each UART includes the following features.

- 5–8 data bits
- Programmable extra stop bit and programmable extra half-stop bit
- Even, odd, and sticky parity bit options
- Additional 8-stage receive FIFO with programmable threshold interrupt
- Flexible transmit and receive interrupt timing
- 3 interrupt outputs for receive, transmit, and status
- Independent DMA operation for receive and transmit

- Programmable automatic RTS/CTS hardware flow control
- False start bit detection
- SIR IrDA operation mode
- MDB/ICP v2.0 operation mode
- Internal loop back
- Improved bit rate granularity
- LIN break command/Inter-frame gap transmission support

**Table 19-1:** UART Specifications

| Feature                  | Availability      |
|--------------------------|-------------------|
| Protocol                 |                   |
| Master-Capable           | Yes               |
| Slave-Capable            | Yes               |
| Transmission Simplex     | Yes               |
| Transmission Half-Duplex | Yes               |
| Transmission Full-Duplex | Yes               |
| Access Type              |                   |
| Data Buffer              | Yes               |
| Core Data Access         | Yes               |
| DMA Data Access          | Yes               |
| DMA Channels             | 2 (per UART Port) |
| DMA Descriptor           | Yes               |
| Boot Capable             | Yes (Slave Mode)  |
| Local Memory             | No                |
| Clock Operation          | SCLK/16           |

## UART Functional Description

The following sections provide details on the UART's functionality.

### UART Register List

The universal asynchronous receiver/transmitter (UART) module is a full-duplex peripheral compatible with PC-style industry-standard UARTs. The UARTs convert data between serial and parallel formats. The serial communication follows an asynchronous protocol that supports various word length, stop bits, and parity generation options. The UARTs include interrupt-handling hardware. Interrupts can be generated from multiple events. A set of registers govern UART operations. For more information on UART functionality, see the UART register descriptions.

**Table 19-2:** UART Module Registers

| Name          | Description                            |
|---------------|----------------------------------------|
| UART_REVID    | Revision ID Register                   |
| UART_CTL      | Control Register                       |
| UART_STAT     | Status Register                        |
| UART_SCR      | Scratch Register                       |
| UART_CLK      | Clock Rate Register                    |
| UART_IMSK     | Interrupt Mask Register                |
| UART_IMSK_SET | Interrupt Mask Set Register            |
| UART_IMSK_CLR | Interrupt Mask Clear Register          |
| UART_RBR      | Receive Buffer Register                |
| UART_THR      | Transmit Hold Register                 |
| UART_TAIP     | Transmit Address/Insert Pulse Register |
| UART_TSR      | Transmit Shift Register                |
| UART_RSR      | Receive Shift Register                 |
| UART_TXCNT    | Transmit Counter Register              |
| UART_RXCNT    | Receive Counter Register               |

## ADSP-BF60x UART Interrupt List

**Table 19-3:** ADSP-BF60x UART Interrupt List

| Description        | ID | DMA Channel | Sensitivity |
|--------------------|----|-------------|-------------|
| UART0 Transmit DMA | 80 | 17          | LEVEL       |
| UART0 Receive DMA  | 81 | 18          | LEVEL       |
| UART0 Status       | 82 |             | LEVEL       |
| UART1 Transmit DMA | 83 | 19          | LEVEL       |
| UART1 Receive DMA  | 84 | 20          | LEVEL       |
| UART1 Status       | 85 |             | LEVEL       |

## ADSP-BF60x UART Trigger List

**Table 19-4:** Masters ADSP-BF60x UART Trigger List

| Description        | ID | Sensitivity |
|--------------------|----|-------------|
| UART0 Transmit DMA | 39 | PULSE/EDGE  |
| UART0 Receive DMA  | 40 | PULSE/EDGE  |
| UART1 Transmit DMA | 41 | PULSE/EDGE  |
| UART1 Receive DMA  | 42 | PULSE/EDGE  |
|                    |    |             |

**Table 19-5:** Slaves ADSP-BF60x UART Trigger List

| Description        | ID | Sensitivity |
|--------------------|----|-------------|
| UART0 Transmit DMA | 39 |             |
| UART0 Receive DMA  | 40 |             |
| UART1 Transmit DMA | 41 |             |



**Table 19-5:** Slaves ADSP-BF60x UART Trigger List (Continued)

| Description       | ID | Sensitivity |
|-------------------|----|-------------|
| UART1 Receive DMA | 42 |             |
|                   |    |             |

## UART Block Diagram

The following figure shows a simplified block diagram of one UART module and how it interconnects to the processor system.

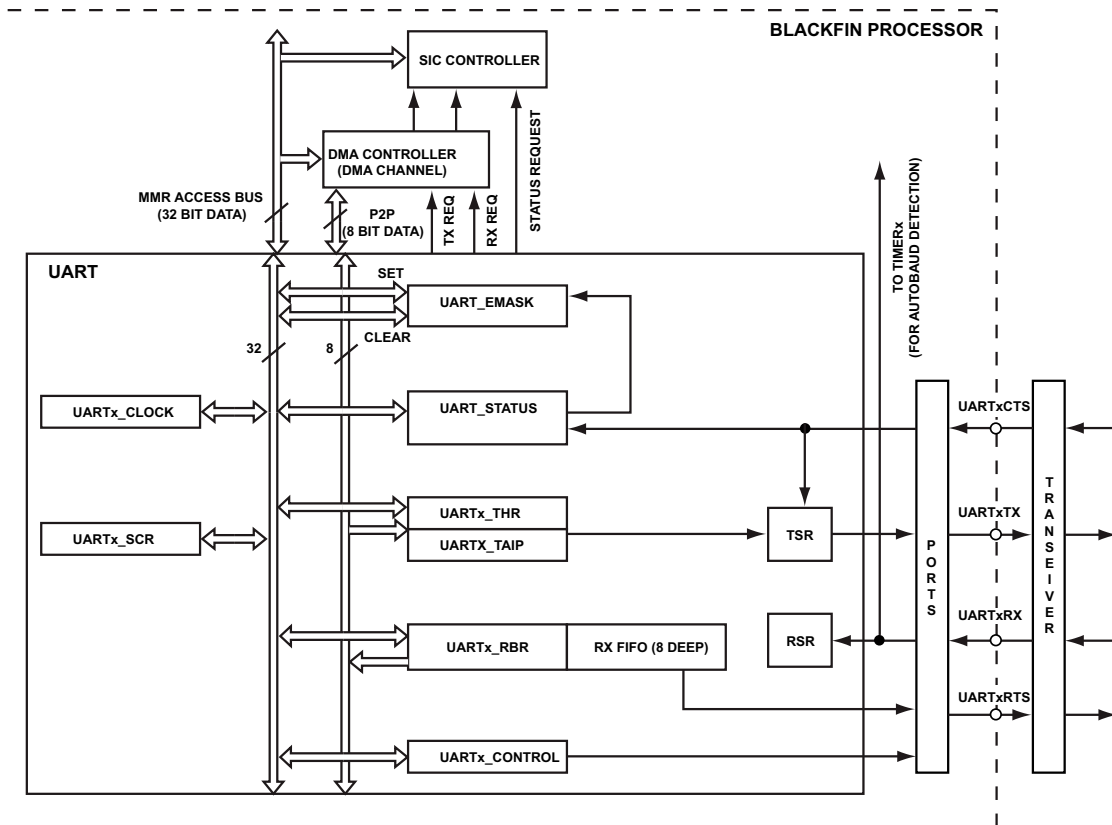


**Figure 19-1:** UART Block Diagram

## UART Architectural Concepts

The following sections provide information about the UART architecture.

## Internal Interface

The UART is a DMA-capable peripheral with support for separate TX and RX DMA master channels. It can be used in either DMA or programmed core modes of operation. The core mode requires software management of the data flow using either interrupts or polling. The DMA method requires minimal software intervention, as the DMA engine itself moves the data.

All UART registers are 32 bits wide. They connect to the peripheral MMR bus. All the bits of all the MMRs may not be used; unused bits are zero-filled. The `UART_RBR` and `UART_THR` registers also connect to one of the P2P buses (8-bit data width).

Each UART has three interrupt outputs. The transmit request and receive request outputs can function as DMA requests and connect to the DMA controller. Therefore, if the DMA is not enabled, the DMA controller simply forwards the request to the system event controller (SEC). The status interrupt output connects directly to the SEC. On many processors, the `UART_RX` pin is also sensed by the alternative capture input (`TMR_AUX_IN`) of one of the GP timers. When configured in capture mode, the GP Timer can then be used to detect the bit rate of the received signal.

## External Interface

Each UART features an `UART_RX` (receive) and an `UART_TX` (transmit) pin available through the general-purpose ports. These two pins usually connect to an external transceiver device that meets the electrical requirements of full duplex (for example, EIA-232, EIA-422, 4-wire EIA-485) or half duplex (for example, 2-wire EIA-485, LIN) standards. Additionally, the UART features a pair of `UART_CTS` (clear to send, input) and `UART_RTS` (request to send, output) pins for hardware flow control. UART signals are usually multiplexed with other functions at the pin level.

## Hardware Flow Control

To prevent the UART transmitter from sending data while the receiving counterpart is not ready, a RTS/CTS hardware flow control mechanism is supported. The RTS signal is an output that connects to the communication partner's CTS input. If data transfer is bidirectional, the handshake is as shown below.

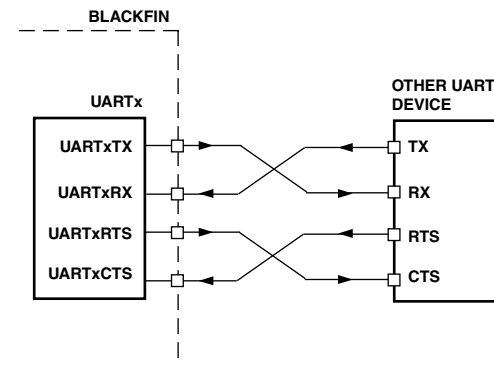


**Figure 19-2:** UART Hardware Flow

Regardless of using DMA or core mode, the receiver can de-assert the `RTS` signal to indicate that its receive buffer is getting full. Further data may cause an overrun error. Consequently, the transmitter pauses when the `CTS` input is in a de-asserted state. If the `UART_CTS` pin is de-asserted, the transmitter still completes transmission of the data currently held in the `UART_TSR` register, but it does not continue with the data in the `UART_THR` register. If the `UART_CTS` pin is asserted again, the transmitter resumes and loads the content of `UART_THR` register into the `UART_TSR` register.

## UART Bit Rate Generation

The sample clock is characterized by the peripheral clock (`SCLK0`) and the 16-bit divisor in the `UART_CLK` register. The UART clock is enabled by the `UART_CTL.EN` bit. By default every serial bit is oversampled 16 times. The bit clock is 1/16th of the sample clock. If not in IrDA mode, the bit clock can equal the sample clock if the `UART_CLK.EDBO` bit is set, so that the following applies:

$$BIT\ RATE = \frac{SCLK0}{16^{(1-EDBO)} \times Divisor}$$

**Figure 19-3:** Bit Rate Equation

The following table provides example divide factors required to support standard baud rates at a `SCLK0` of 125 MHz.

**Table 19-6:** UART Bit Rate Examples With 125 MHz `SCLK0`

| Bit Rate  | D factor = 16 |              |         | D factor = 1 |              |         |
|-----------|---------------|--------------|---------|--------------|--------------|---------|
|           | DL            | Actual       | % Error | DL           | Actual       | % Error |
| 2,400     | 3,255         | 2,400.15     | 0.006   | 52,083       | 2,400.02     | 0.001   |
| 4,800     | 1,628         | 4,798.83     | 0.024   | 26,042       | 4,799.94     | 0.001   |
| 9,600     | 814           | 9,597.67     | 0.024   | 13,021       | 9,599.88     | 0.001   |
| 19,200    | 407           | 19,195.33    | 0.024   | 6,510        | 19,201.23    | 0.006   |
| 38,400    | 203           | 38,485.22    | 0.022   | 3,255        | 38,402.46    | 0.006   |
| 57,600    | 136           | 57,444.85    | 0.269   | 2,170        | 57,603.69    | 0.006   |
| 115,200   | 68            | 114,889.71   | 0.269   | 1,085        | 115,207.37   | 0.006   |
| 921,600   | 8             | 976,562.50   | 5.964   | 136          | 919,117.65   | 0.269   |
| 1,500,000 | 5             | 1,562,500.00 | 4.167   | 83           | 1,516,021.10 | 0.402   |

**Table 19-6:** UART Bit Rate Examples With 125 MHz SCLK0 (Continued)

| Bit Rate  | D factor = 16 |              |         | D factor = 1 |              |         |
|-----------|---------------|--------------|---------|--------------|--------------|---------|
|           | DL            | Actual       | % Error | DL           | Actual       | % Error |
| 3,000,000 | 3             | 2,604,166.67 | 13.194  | 42           | 2,976,190.48 | 0.794   |
| 6,250,000 | 1             | 7,812,500.00 | 25.000  | 20           | 6,250,000.00 | 0.000   |

**NOTE:** Careful selection of SCLK0 frequencies—that is, even multiples of desired bit rates— can result in lower error percentages.

Setting the bit clock equal to the sample clock (UART\_CLK.EDB0 =1) improves bit rate granularity and enables the bit clock to more closely match the bit rate of the communication partner. There is, however, a disadvantage—the power dissipation is higher. Also the sample points may not be that accurate. Therefore, it is recommended to use UART\_CLK.EDB0 =1 mode only when bit rate accuracy is not acceptable in UART\_CLK.EDB0 =0 mode.

The UART\_CLK.EDB0 =1 mode is not intended to increase operation speed beyond the electrical limitations of the UART transfer protocol.

## Autobaud Detection

At the chip level, the UART RX pin is usually routed to an alternate capture input (TMR\_AUX\_IN) of a general purpose timer. When working in Width Capture mode, this general-purpose timer can be used to automatically detect the bit rate applied to the RX pin by an external device. The capture capabilities of the timer are often used to supervise the bit rate at runtime. If the UART was talking to any device supplied by a weak clock oscillator that drifts over time, the processor can then readjust its UART bit rate dynamically as required.

Often, autobaud detection is used for initial bit rate negotiations. There, the processor is most likely a slave device waiting for the host to send a predefined autobaud character. This is a common scenario for UART booting. In this scenario, it is recommended that the UART\_CTL.EN bit is not enabled while autobaud detection is performed, to prevent the UART from starting reception with incorrect bit rate matching. Alternatively, the UART can be disconnected from its RX pin by setting the UART\_CTL.LOOP\_EN bit.

A software routine can detect the pulse widths of serial stream bit cells. Because the sample base of the timer is synchronous with the UART operation—all derived from the same SCLK0—the pulse widths can be used to calculate the bit rate divider for the UART by using the following formula.

$$DIVISOR = \frac{TIMER\_TMR\_WID}{16^{(1-EDB0)} \times \text{Number of captured UART bits}}$$

**Figure 19-4:** UART Bit Rate Divider Calculation

In order to increase the number of timer counts and therefore the resolution of the captured signal, it is recommended not to measure just the pulse width of a single bit, but to enlarge the pulse of interest over more bits. Traditionally, a NULL character (ASCII 0x00) is used in autobaud detection, as shown below.

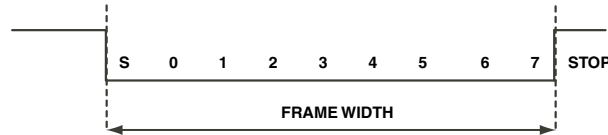


**Figure 19-5:** Autobaud Detection

Because the example frame encloses 8 data bits and 1 start bit, apply the following formula.

$$DIVISOR = \frac{TIMER\_TMR\_WID}{16^{(1-EDB0)} \times 9}$$

**Figure 19-6:** Autobaud Detection (8 Data Bits, 1 Start Bit)

Real RX signals often have asymmetrical falling and rising edges, and the sampling logic level is not exactly in the middle of the signal voltage range. At higher bit rates, such pulse-width-based autobaud detection might not return adequate results without additional analog signal conditioning. Measuring signal periods works around this issue and is strongly recommended.

For example, predefine ASCII character “@” (0x40) as the autobaud detection character and measure the period between two subsequent falling edges. As shown in the figure below, measure the period between the falling edge of the start bit and the falling edge after bit 6. Since this period encloses 8 bits, apply the following formula.

$$DIVISOR = \frac{TIMER\_TMR\_PER}{16^{(1-EDB0)} \times 8}$$

**Figure 19-7:** Autobaud Detection (8 Bits)

Or:

- Divisor = `TIMER_TMR_PER >> 7` if `UART_CLK.EDB0 = 0`
- Divisor = `TIMER_TMR_PER >> 3` if `UART_CLK.EDB0 = 1`

The following figure shows the ASCII “@” (0x40) detection character.

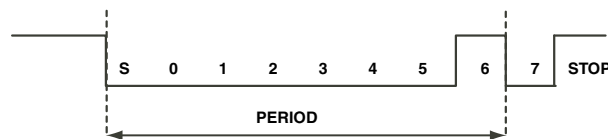


**Figure 19-8:** Autobaud Detection Character 0x40

## UART Debug Features

The UART has the option to automatically calculate and transmit a parity bit. The following table summarizes parity behavior assuming 8-bit data words (`UART_CTL.WLS = b#11`).

**Table 19-7:** UART Parity

| PEN | STP | EPS | Data (hex) | Data (binary, LSB first) | Parity |
|-----|-----|-----|------------|--------------------------|--------|
| 0   | x   | x   | x          | x                        | None   |
| 1   | 0   | 0   | 0x60       | 0000 0110                | 1      |
| 1   | 0   | 0   | 0x57       | 1110 1010                | 0      |
| 1   | 0   | 1   | 0x60       | 0000 0110                | 0      |
| 1   | 0   | 1   | 0x57       | 1110 1010                | 1      |
| 1   | 1   | 0   | x          | x                        | 1      |
| 1   | 1   | 0   | x          | x                        | 1      |
| 1   | 1   | 1   | x          | x                        | 0      |
| 1   | 1   | 1   | x          | x                        | 0      |

The two force error bits, `UART_CTL.FPE` and `UART_CTL.FFE`, are intended for test purposes. They are useful for debugging software, especially in loop back mode.

The UART can be set to internal loop back mode (`UART_CTL.LOOP_EN = 1`). Loop back mode disconnects the receiver's input from the RX pin and internally redirects the transmit output to the receiver. The TX pin remains active and continues to transmit data externally as well. Loop back mode also forces the `UART_RTS` pin to de-assert, disconnects the CTS bit from the `UART_CTS` input pin, and connects the internal version of `UART_RTS` to the `UART_STAT.CTS` bit.

Additionally, the TX pin can be forced to zero asynchronously through the `UART_CTL.SB` bit.

## UART Operating Modes

The UART's main operating modes are UART Mode, IrDA SIR Mode, and MDB Mode.

## UART Mode

The UART Mode follows an asynchronous serial communication protocol with these options:

- 1 start bit
- 5-8 data bits
- Address bit
- None, even, odd or sticky parity
- 1, 1½, or 2 stop bits (1½ stop bits valid only in 5-bit word length)

The format of received and transmitted character frames is controlled by the `UART_CTL` register. Data is always transmitted and received with the least significant bit (LSB) first.

The following figure shows a typical physical bit stream measured on a TX pin.

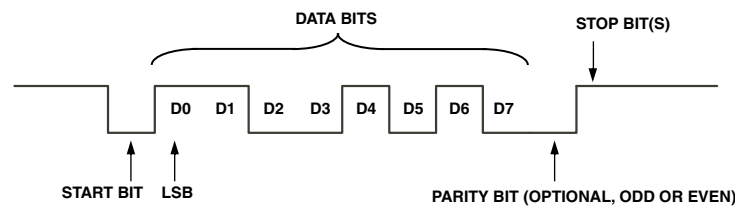


**Figure 19-9:** Bit stream on a UART TX Pin Transmitting an “S” Character (0x53)

## IrDA SIR Mode

Aside from the standard UART functionality, the UART also supports serial data communication by way of infrared signals, according to the recommendations of the Infrared Data Association (IrDA). The physical layer known as IrDA SIR (9.6/115.2 Kbps rate) is based on return-to-zero-inverted (RZI) modulation. Pulse position modulation is not supported.

Using the 16x data rate clock, RZI modulation is achieved by inverting and modulating the non-return-to-zero (NRZ) code normally transmitted by the UART. On the receive side, the 16x clock is used to determine an IrDA pulse sample window, from which the RZI modulated NRZ code is recovered.

**NOTE:** The `UART_CLK.EDB0` bit is not valid in IrDA mode—this bit should be cleared (= 0) in this mode.

## MDB Mode

The Multi-Drop Bus (MDB) mode is very similar to the UART mode, but also includes support for an Address (or Mode) bit. With the addition of the Address bit, the UART can provide hardware support for the Multi-Drop Bus/Internal Communication Protocol (MDB/ICP v2.0).

In multi-drop bus mode, the address bit is used to indicate if the transferred word is an address word or a data word. An address bit of 1 signifies an address word and requires all slave devices to read the word. An address bit of 0 signifies a data word and only the slave that is being addressed needs to read the word.

The address bit is similar to the parity bit in the sense that it is not part of the data word, but is appended to the transmitted data word by the hardware. Therefore, a write to the `UART_THR/UART_TAIP` registers requires writing only the data word. A read from `UART_RBR` returns only the data word, the `UART_STAT.ADDR` bit contains the address bit corresponding to the word in the `UART_RBR` register.

The MDB mode follows an asynchronous serial communication protocol with the following options.

- 1 start bit
- 5-8 data bits
- Address bit
- None, even, odd or sticky parity
- 1, 1½, or 2 stop bits (1½ stop bits valid only in 5-bit word length)

**NOTE:** If the address bit and parity bit are both enabled, the parity check and generation includes the address bit in its parity calculation.

## UART Data Transfer Modes

The UART is capable of transferring data using both the core and DMA. Receive and transmit paths operate completely independently except that the bit rate and the frame format are identical for both transfer directions. Transmit and receive channels are both buffered. The `UART_THR` register buffers the transmit shift register (`UART_TSR`) and the `UART_RBR` register buffers the receive shift register (`UART_RSR`).

## UART Mode Transmit Operation (Core)

In core mode, data is moved to and from the UART by the processor core. A write to the `UART_THR` register initiates the transmit operation. If no former operation is pending, the data is immediately passed from the `UART_THR` register to the `UART_TSR` register. There, it is shifted out at the bit rate characterized by the `UART_CTL` register, with start, stop, and parity bits appended as defined by the `UART_CTL` register. The `UART_THR` register and the `UART_TSR` register can be modeled as a two-stage transmit buffer. The least significant bit (LSB) is always transmitted first. This is bit 0 of the value written to `UART_THR`.

## UART Mode Break Command and Inter-Frame Gap Transmit Operation

Break Command/Inter-frame gap functionality is available only in UART mode. The `UART_TAIP` register has different functionality in other modes. Break command/Inter-frame gap transmission is initiated by writes to the `UART_TAIP` register. If no former transmit operation is pending (in the `UART_TSR` register), a pulse is transmitted for the duration specified by `UART_TAIP[6-0]`/bit rate, where bit rate is set by the `UART_CLK` register. The polarity of this pulse is set by bit 7 in the `UART_TAIP` register.



**NOTE:** The `UART_CTL.TPOLC` setting, if enabled, could lead to an inverted most-significant bit being transmitted.

**NOTE:** If other transmission is pending (in the `UART_TSR` register), the `UART_TAIP` initiated pulse is queued until after all pending operations have finished and all stop bits are transmitted.

The transmission of break command/inter-frame gap is followed by transmission of the number of stop bits as set in the `UART_CTL.STB` and `UART_CTL.STBH` bit fields.

## UART Mode Receive Operation (Core)

The receive operation uses the same data format as the transmit configuration, except that one valid stop bit is always sufficient; that is, the `STB` bit and `STBH` bit have no impact to the receiver.

The UART receiver senses the falling edges of the RX input. When an edge is detected, the receiver starts sampling the input according to settings in the `UART_CLK` register. The start bit is sampled (majority sampling) close to its midpoint. If sampled low, a valid start condition is assumed. Otherwise, the detected falling edge is discarded.

After detection of the start bit, the received word is shifted into the `UART_RSR` register.

After the corresponding stop bit is received, the content of the `UART_RSR` register is transferred to the 8-deep receive FIFO and is accessible by reading the `UART_RBR` register.

The receive FIFOs and the `UART_RBR` register can be seen as a 9-stage receive buffer. If the stop bit of the 9th word is received before software reads the `UART_RBR` register, an overrun error is reported. The overrun case protects data in the `UART_RBR` and the receive FIFO from being overwritten by further data until the OE bit is cleared by software. The data in the `UART_RSR` register, however, is immediately destroyed as soon as the overrun occurs.

The sampling clock is 16 times faster than the bit clock. The receiver over samples every bit 16 times and makes a majority decision based on the middle three samples. This improves immunity against noise and hazards on the line. Spurious pulses of less than two times the sampling clock period are disregarded.

Normally, every incoming bit is sampled at exactly the 7th, 8th and 9th sample clock. If, however, the `UART_CLK.EDBO` bit is set to 1 to achieve better bit rate granularity and accuracy as required at high operation speeds, the bits are one roughly sampled at 7/16th, 8/16th and 9/16th of their period. Hardware design should ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

Reception is started when a falling edge is detected on the RX input pin. The receiver attempts to see a start bit. The data is shifted into the `UART_RSR` register. After the 9th sample of the first stop bit is processed, the received data is copied to the 8-stage receive FIFO and the `UART_RSR` recovers for further data reception.

The receiver samples data bits close to their midpoint. Because the receiver clock is usually asynchronous to the transmitter's data rate, the sampling point may drift relative to the center of the data bits. The sampling point is synchronized again with each start bit, so the error accumulates only over the length of a single word. The polarity of received data is selectable, using the `UART_CTL.RPOLC` bit.

**NOTE:** The receiver checks for only a single stop bit. After the third sample of the first stop bit has been received (at time 9/16th of the stop bit duration), the receiver immediately takes action (status update) and prepares itself for new falling edge detection (start detection).

## IrDA Transmit Operation

To generate the IrDA pulse transmitted by the UART, the normal NRZ output of the transmitter is first inverted if the `UART_CTL.TPOLC` bit is configured for active-low operation, such that a 0 is transmitted as a high pulse of 16 UART clock periods and a 1 is transmitted as a low pulse for 16 UART clock periods. The leading edge of the pulse is then delayed by six UART clock periods. Similarly, the trailing edge of the pulse is truncated by eight UART clock periods. For a 16-cycle UART clock period, this results in the final representation of the original 0 as a high pulse of only 3/16 clock periods. The pulse is centered around the middle of the bit time, as shown in the figure below. The final IrDA pulse is fed to the off-chip infrared driver.

This modulation approach ensures a pulse width output from the UART of three cycles high out of every 16 UART clock cycles. As shown in the figure below, the error terms associated with the bit rate generator are very small and well within the tolerance of most infrared transceiver specifications.

**NOTE:** In IrDA mode, writes to the `UART_TAIP` register are equivalent to writes to the `UART_THR` register.

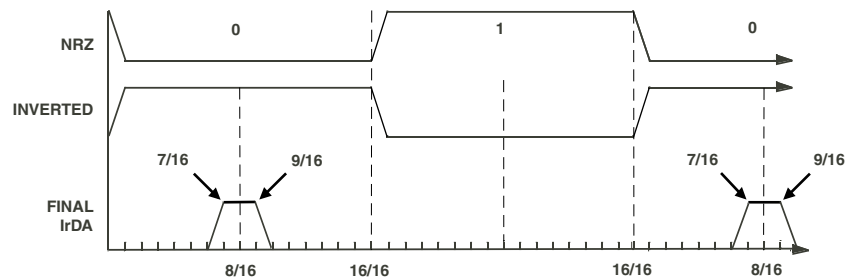


Figure 19-10: IrDA Transmit Pulse

## IrDA Receive Operation

The IrDA receiver function is more complex than the transmit function. The receiver must discriminate the IrDA pulse and reject noise. To do this, the receiver looks for the IrDA pulse in a narrow window centered around the middle of the expected pulse.

Glitch filtering is accomplished by counting 16 system clocks from the time an initial pulse is seen. If the pulse is absent when the counter expires, it is considered a glitch. Otherwise, it is interpreted as a 0. This is acceptable because glitches originating from on-chip capacitive cross-coupling typically do not last for more than a fraction of the system clock (SCLK0) period. Sources outside of the chip and not part of the transmitter can be avoided by appropriate shielding. The only other source of a glitch is the transmitter itself. The processor relies on the transmitter to perform within specification. If the transmitter violates the specification, unpredictable results may occur. The 4-bit counter adds an extra level of protection at a minimal cost.

**NOTE:** Because SCLK0 can change across systems, the longest glitch tolerated is inversely proportional to the SCLK0 frequency.

The receive sampling window is determined by a counter that is clocked at the 16x bit-time sample clock. The sampling window is re-synchronized with each start bit by centering the sampling window around the start bit.

The polarity of receive data is selectable, using the `UART_CTL.RPOLC` bit. The following figure provides examples of each polarity type.

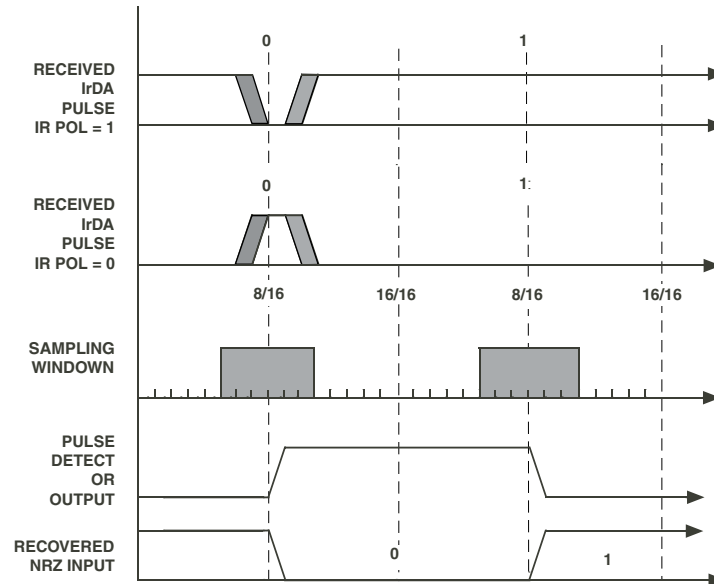


**Figure 19-11:** IrDA Receiver Pulse Detection

## MDB Transmit Operation

In MDB mode, receive and transmit paths operate completely independently from each other, except for sharing bit rate and frame formats for both transfer directions.

Transmit operation is initiated by writing the `UART_THR` or `UART_TAIP` registers. A write to the `UART_THR` register transmits the written word with the appending address bit set low, a write to the `UART_TAIP` register transmits the written word with the appended address bit set high. The data is moved into the `UART_TSR` register, where it is shifted out at the bit rate programmed by the `UART_CLK` register, with start, stop, address, and parity bits appended as required.

If DMA is enabled, the DMA engine always writes the data into the `UART_THR` register, and the written word is transmitted with the appending address bit set low.

The polarity of transmit data is selectable, using the `UART_CTL.TPOLC` bit.

## MDB Receive Operation

Receive operations use the same data format as the transmit configuration, except that the number of stop bits is always assumed to be 1. After detection of the start bit, the received word is shifted into the `UART_RSR` register at the programmed bit.

Normally, every incoming bit is sampled at exactly the 7th, 8th and 9th sample clock. If, however, the `UART_CLK.EDBO` bit is set to achieve better bit rate granularity and accuracy as required at high operation speeds, the bits are roughly sampled at 7/16th, 8/16th and 9/16th of their period. Hardware design should ensure that the incoming signal is stable between 6/16th and 10/16th of the nominal bit period.

After the appropriate number of bits (including address, parity, and stop bits) is received, the `UART_RSR` register is transferred to the receive FIFO and accessible through the `UART_RBR` register.

The polarity of receive data is selectable, using the `UART_CTL.RPOLC` bit.

## DMA Modes

In DMA modes, separate receive and transmit DMA channels move data between the UART and memory. The software does not have to move data; it just has to set up the appropriate transfers either through the descriptor mechanism or through autobuffer mode.

DMA channels provide a 4-deep FIFO, resulting in total buffer capabilities of 6 words at the transmit side and 9 words at the receive side. In DMA mode, the latency is determined by the bus activity and arbitration mechanism and not by the processor loading and interrupt priorities.

To enable UART DMA, first set up the system DMA control registers and then enable the `UART_IMSK.ERBFI` and/or `UART_IMSK.ETBEI` interrupts. This is necessary because these interrupt request lines double as DMA request lines. With DMA enabled, once these requests are received, the DMA control unit generates a direct memory access. If DMA is not enabled, the UART interrupt is passed on to the system interrupt handling unit.

**NOTE:** The UART's status interrupt goes directly to the system interrupt handling unit, bypassing the DMA unit completely.

For transmit DMA, programs should set the `SYNC` bit in the `DMAx_CFG` register. With this bit set, interrupt generation is delayed until the entire DMA FIFO is drained to the UART module. The UART TX DMA interrupt service routine is allowed to disable the DMA or to clear the `ETBEI` control bit only when the `SYNC` bit is set, otherwise up to four data bytes might be lost.

When the `ETBEI` bit is set in the `UART_IMSK` register, an initial transmit DMA request is issued immediately. The program should then clear the `ETBEI` bit through the DMA service routine.

In DMA transmit mode, the `ETBEI` bit enables the peripheral request to the DMA FIFO. The strobe on the memory side is still enabled by the `DMA_CFG.EN` bit. If the DMA count is less than the DMA FIFO depth, which is 4, then the DMA interrupt might be requested before the `ETBEI` bit is set. If this behavior is not wanted, set the `SYNC` bit in the `DMAx_CONFIG` register.

Regardless of the `DMAx_CONFIG.SYNC` setting, the DMA stream has not left the UART transmitter completely at the time the interrupt is generated. Transmission may abort in the middle of the stream, causing data loss, if the UART clock was disabled without additional synchronization with the `TEMT` bit.

The UART provides functionality to avoid resource consuming polling of the `TEMT` bit. The `EDTPTI` bit in the `UART_IMSK_SET` register enables the `TEMT` bit to trigger a DMA interrupt. To delay the DMA completion interrupt until the last data word of a STOP DMA has left the UART, keep the `DMA_CFG.DI_EN` bit cleared and set the `EDTPTI` bit instead. Then, the normal DMA completion interrupt is suppressed. Later, the `TEMT` event triggers a DMA interrupt after the DMA's last word has left the UART transmit buffers. If `DI_EN` and `EDTPTI` are set, when finishing STOP mode, the DMA requests two interrupts.

The UART's DMA supports 8-bit and 16-bit operation, but not 32-bit operation. Sign extension is not supported.

## Mixing DMA and Core Modes

Especially on the transmit side, switching from DMA mode to core operation on the fly requires some consideration. By default, the interrupt timing of the DMA is synchronized with the memory side of the DMA FIFOs. Normally, the TX DMA completion interrupt is generated after the last byte is copied from the memory into the DMA FIFO. The TX DMA interrupt service routine is not yet permitted to disable the `DMA_CFG.EN` bit. The interrupt is requested by the time the `DMA_STAT.DONE` bit is set. The `DMA_STAT.RUN` bit, however, remains set until the data has completely left the transmit DMA FIFO.

Therefore, when planning to switch from a DMA to the core mode, always set the `SYNC` bit in the `DMA_CFG` word of the last descriptor or work unit before handing over control to core mode. Then, after the interrupt occurs, software can write new data into the `UART_THR` register as soon as the `THRE` bit permits. If the `SYNC` bit cannot be set, software can poll the `DMA_STAT.RUN` bit instead. Using the `EDTPTI` bit can avoid expensive status bit polling, alternatively.

When switching from core to DMA operation, take care that the very first DMA request is issued properly. If the DMA is enabled while the UART is still transmitting, no precaution is required. If, however, the DMA is enabled after the `TEMT` bit is high, the `ETBEI` bit should be pulsed to initiate DMA transmission.

## Setting Up Hardware Flow Control

Use the following steps to setup UART hardware flow control:

1. Configure automatic or manual hardware flow control for the receiver through the `UART_CTL.ARTS` bit, and/or the transmitter through the `UART_CTL.ACTS` bit.
2. Configure CTS and RTS polarity through the `UART_CTL.FCPOL` bit.

*AFTER COMPLETING THIS TASK:*

On reset, when the UART is not yet enabled and the port multiplexing has not been programmed, the `RTS` pin is not driven. Some applications may require the `RTS` signal to be pulled to either state by a resistor during reset.

## UART Event Control

Status flags in the `UART_STAT` register are available to signal data reception, parity, and error conditions, if required.

### Interrupt Masks

Each UART features a set of Interrupt Mask Registers: `UART_IMSK`, `UART_IMSK_SET`, and `UART_IMSK_CLR`. The `UART_IMSK` register supports read/write operations. Writing ones to the `UART_IMSK_SET` register enables interrupts, writing ones to the `UART_IMSK_CLR` register disables them. Reads from either register return the enabled bits. This way, different interrupt service routines can control transmit, receive, and status interrupts independently and easily.

The `UART_IMSK` registers are used to enable requests for system handling of empty or full states of UART data registers. Unless polling is used as a means of action, the `UART_IMSK.ERBFI` and/or `UART_IMSK.ETBEI` bits in this register are normally set.

Each UART module has three interrupt outputs. One is dedicated for transmission, one for reception, and the third is used to report status events. Transmit and receive requests are routed through the DMA controller. The status request goes directly to the system event controller (SEC).

If the associated DMA channel is enabled, the request functions as a DMA request. If the DMA channel is disabled, it simply forwards the request to the SEC. Note that a DMA channel must be associated with the UART module to enable TX and RX interrupts. Otherwise, transmit and receive requests cannot be forwarded.

**NOTE:** To operate in interrupt mode without using DMA channels, set the `UART_IMSK.ELSI` bit. This redirects receive and transmit requests to the status interrupt output. The status interrupt goes directly to the SEC without being routed through the DMA controller

### Interrupt Servicing

UART writes and reads can be accomplished through interrupt service routines (ISRs). Separate interrupt lines are provided for TX, RX, and status. The independent interrupts can be enabled individually by the `UART_IMSK` register group. The `UART_CTL.EN` bit must be set to enable UART transmit interrupts.

The ISRs can evaluate the status bits in the `UART_STAT` register to determine the signaling interrupt source. Interrupts must also be assigned and unmasked by the processor's system event controller. The ISRs must clear the interrupt latches explicitly. To reduce interrupt frequency on the receive side in core mode, the `UART_IMSK.ERFCI` status interrupt may be used as an alternative to the regular `UART_IMSK.ERBFI` receive interrupt. Hardware must ensure that at least two (if `UART_CTL.RFIT=0`) or four (if `UART_CTL.RFIT=1`) words are available in the receive buffer by the time the interrupt is requested.

## Transmit Interrupts

Transmit interrupts are enabled by the `UART_IMSK_SET.ETBEI` bit.

The `UARTx_THR` and `UARTx_TAIP` registers are the same physical register, and both affect the signaling of the `UART_STAT.TEMT`, `UART_STAT.TFI`, and `UART_STAT.THRE` bits similarly.

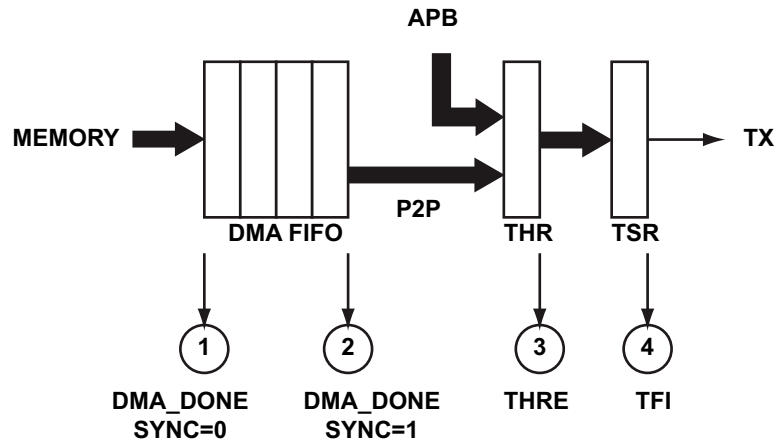


**Figure 19-12:** Transmit Interrupts

The transmit request is asserted along with the `UART_STAT.THRE` bit, indicating that the TX buffer is ready for new data. Note that the `UART_STAT.THRE` bit resets to 1. When the `UART_IMSK_SET.ETBEI` bit is set, the UART module immediately issues an interrupt or DMA request. This way, no special handling of the first character is required when transmission of a string is initiated. Simply set the `UART_IMSK_SET.ETBEI` bit and let the interrupt service routine load the first character from memory and write it to the `UART_THR` register in the normal manner. Accordingly, the `UART_IMSK.ETBEI` bit can be cleared through the `UART_IMSK_CLR` register if the string transmission has completed.

The `UART_STAT.THRE` bit is cleared by hardware when new data is written to the `UARTx_THR` register. These writes also clear the TX interrupt request. However, they also initiate further transmission. If software doesn't want to continue transmission, the TX request can alternatively be cleared through the `UART_IMSK_CLR.ETBEI` bit register. Transfers of data from the `UART_THR` register to the `UART_TSR` register set this status flag in the `UART_STAT` register again.

The `UART_STAT.TEMT` bit can be interrogated to see whether any transmission is ongoing. The `UART_STAT.TEMT` bit's sticky counterpart, `UART_STAT.TFI`, indicates if the transmit buffer has drained and can trigger a status interrupt, if required. When data is pending in either one of these registers, the `UART_STAT.TEMT` flag is low. As soon as all data has left the `UART_TSR` register, the `UART_STAT.TEMT` bit goes high again and indicates that all pending transmit operation (including stop bits) has finished. At that time it is safe to disable the `UART_CTL.EN` bit or to three-state off-chip line drivers. An interrupt can be generated by that time either through the status interrupt channel when the `UART_IMSK.ETFI` bit is set, or through the DMA controller when enabled by the `UART_IMSK.EDTPTI` bit.

When enabled by the `UART_IMSK.ETBEI` bit, the `UART_STAT.THRE` flag requests data along the peripheral command lines to the DMA controller (hereafter referred to as TXREQ). This signal is routed through the

DMA controller. If the associated DMA channel is enabled, the TXREQ signal functions as a DMA request, otherwise the DMA controller simply forwards it to the SEC. Alternatively the UART\_IMSK.ETXS bit can redirect the transmit interrupts to the UART status interrupt.

With interrupts disabled, these status flags can be polled to determine when data is ready to move. Note that because polling is processor intensive, it is not typically used in real-time signal processing environments. Since read operations from UART\_STAT registers have no side effects, different software threads can interrogate these registers without mutual impacts. Polling the SEC\_SSTATn register without enabling the interrupts by the SEC\_CCTL register is an alternate method of operation to consider. Software can write up to two words into the UART\_THR register before enabling the UART clock. As soon as the UART\_CTL.EN bit is set, those two words are sent.

## Receive Interrupts

Receive interrupts are enabled by the UART\_IMSK\_SET.ERBFI bit. If set, the UART\_STAT.DR flag requests an interrupt on the dedicated RXREQ output, indicating that new data is available in the UART\_RBR register. This signal is routed through the DMA controller. If the associated DMA channel is enabled, the RXREQ signal functions as a DMA request; otherwise the DMA controller simply forwards it to the SEC. Alternatively, if no DMA channel is assigned to the UART, the UART\_IMSK.ERXS bit can redirect the receive interrupts to the UART status interrupt. When software reads the UART\_RBR register, hardware clears the UART\_STAT.DR bit again, which, in turn, clears the receive interrupt request.

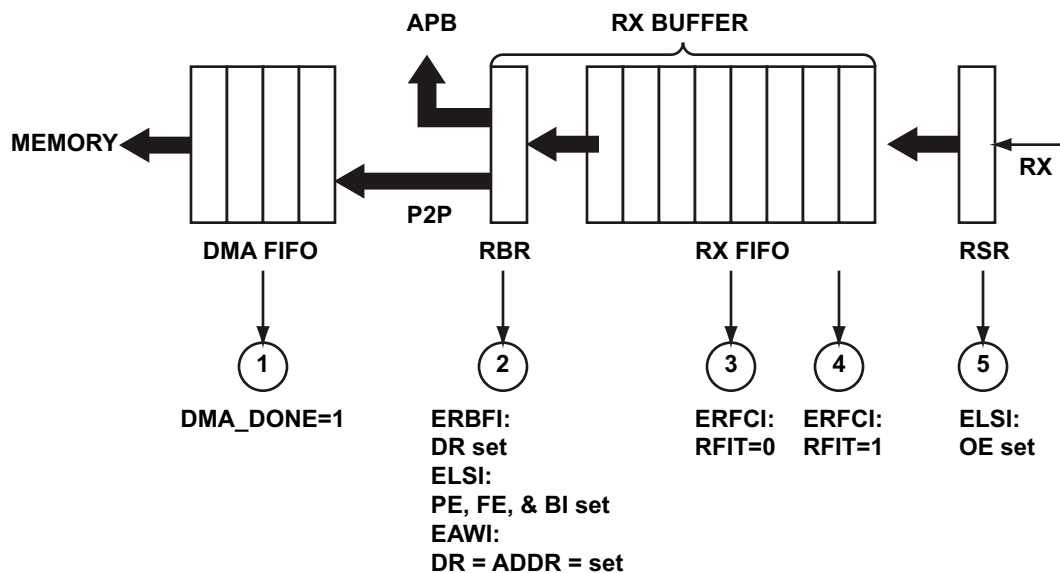


**Figure 19-13:** Receive Interrupts

The UART\_STAT.DR, UART\_STAT.ADDR, UART\_STAT.ASTKY, UART\_STAT.PE, UART\_STAT.FE, and UART\_STAT.BI bits are updated along with UART\_RBR register. The UART\_STAT.OE bit updated as soon as an overflow condition occurs (for example when a frame's stop bit is received and the receive FIFO is full). When the UART\_RBR register is not read in time, the received data is protected from being overwritten by



new data until the `UART_STAT.OE` bit is cleared by software. Only the content of the `UART_RSR` register can be overwritten in the overrun case.

The state of the 8-deep receive FIFO can be monitored by the `UART_STAT.RFCS` bit. The buffer's behavior is controlled by the `UARTx_CTL.RFIT` bit. If `UART_CTL.RFIT` is zero, the `UART_STAT.RFCS` bit is set when the receive buffer holds four or more words. If `UART_CTL.RFIT` is set, the `UART_STAT.RFCS` bit is set when the receive buffer holds seven or more words. The `UART_STAT.RFCS` bit is cleared by hardware when a core or DMA reads the `UART_RBR` register and when the buffer is flushed below the level of four (`UART_CTL.RFIT = 0`) or seven (`UART_CTL.RFIT = 1`). If the associated interrupt bit `UART_IMSK.ERFCI` is enabled, a status interrupt is reported when the `UART_STAT.RFCS` bit is set.

If errors are detected during reception, an interrupt can be requested from the status interrupt output. This status interrupt request goes directly to the SEC. Status interrupt requests are enabled by the `UART_IMSK_SET.ELSI` bit.

The controller detects the following error conditions, shown with their associated bits in the `UART_STAT` register.

- Overrun error (`UART_STAT.OE` bit)
- Parity error (`UART_STAT.PE` bit)
- Framing error/invalid stop bit (`UART_STAT.FE` bit)
- Break indicator (`UART_STAT.BI` bit)

## Status Interrupts

The UART status interrupt channels are used for the following purposes.

- Line status interrupts
- Flow control interrupts
- Receive FIFO threshold interrupts
- Transmission finished interrupt

Line status interrupts are enabled by the `UART_IMSK.ELSI` bit. If set, the status interrupt request is asserted with any of the `UART_STAT.BI`, `UART_STAT.FE`, `UART_STAT.PE`, or `UART_STAT.OE` receive errors bits. The error bits in the `UART_STAT` register are cleared by W1C operation. Once all error conditions are cleared, the interrupt request de-asserts.

The receive FIFO count interrupt is enabled by the `UART_IMSK_SET.ERFCI` bit. If set, a status interrupt is generated when the `UART_STAT.RFCS` is active. The `UART_STAT.RFCS` bit indicates a receive buffer threshold level. If the `UART_CTL.RFIT` bit is cleared, software can safely read two words out of the `UART_RBR` register by the time the `UART_STAT.RFCS` interrupt occurs. If the `UART_CTL.RFIT` bit is set, software can safely read four words. The interrupt and the `UART_STAT.RFCS` bit clear when the `UART_RBR` is read a sufficient number of times, so that the receive buffer drains below the threshold of two

(`UART_CTL.RFIT = 0`) or four (`UART_CTL.RFIT = 1`). Because in DMA mode a status service routine may not be permitted to read `UART_RBR`, this interrupt is only recommended in core mode. In DMA mode, use this functionality for error recovery only.

The flow control interrupts are enabled by the `UART_IMSK_SET.EDSSI` bit. If active, a status interrupt is generated when the sticky `UART_STAT.SCTS` bit register is set, indicating that the transmitter's `UART_CTS` input been re-asserted. A W1C operation to the `SCTS` bit clears the interrupt request.

A transmission finished interrupt is enabled by the `UART_IMSK_SET.ETFI` bit. If active, a status interrupt request is asserted when the `UART_STAT.TFI` bit is set. The `UART_STAT.TFI` is the sticky version of the `UART_STAT.TEMT` bit, indicating that a byte that started transmission has completely finished. The interrupt request is cleared by a W1C operation to the `UART_STAT.TFI` bit.

## Multi-Drop Bus Events

Several status bits and interrupt features in the `UART_STAT` and `UART_IMSK` registers facilitate efficient data handling in multi-drop bus mode. These include the address (`UART_STAT.ADDR`) bit, address sticky (`UART_STAT.ASTKY`) bit and enable address word interrupt (`UART_IMSK.EAWI`). One of the key features of the multi-drop bus protocol is its address bit, which signifies to the slaves that the master is transmitting an address word (to be read by all) or a data word (to be read by the addressed slave only). The UART hardware provides for an efficient method of handling the situation described above with the use of `UART_STAT.ASTKY` bit.

**NOTE:** The `UART_STAT.ASTKY` bit is used in multi-drop bus mode to indicate if a peripheral is currently being addressed. The `UART_STAT.ASTKY` bit is a sticky version of the `UART_STAT.ADDR` bit and is set by hardware whenever the `UART_STAT.ADDR` bit is set. It can only be cleared by software with a W1C operation. With the `ASTKY` bit set, words are received irrespective of the mode bit/address bit setting. With the `UART_STAT.ASTKY` bit cleared, only address words (mode bit = 1) are received and words with mode bit = 0 is ignored (not moved from the `UART_RSR` to the Receive FIFO) in MDB mode. This bit does not affect reception in non-MDB modes.

## UART Programming Model

The following sections provide basic procedures for configuring various UART operations.

### Detecting Autobaud

Please refer to [Autobaud Detection](#) for more information. The required steps are:

1. Ensure that the timer is disabled.
2. Configure the following bits: `UART_CTL.MOD = 00`, `UART_CTL.LOOP_EN = 1`, `UART_CTL.WLS = 11` (8-bit data), and `UART_CTL.EN = 1`

3. Configure the following bits: `TIMER_TMRn_CFG.TMODE = 1101`, `TIMER_TMRn_CFG.OUTDIS = 1`, `TIMER_TMRn_CFG.IRQMODE = 10` and enable the timer.
4. Send test data through the host device and wait for the timer interrupt and disable the timer.
5. The bit rate can be derived from the timer period register value according to the formula provided in the [Autobaud Detection](#) section.

## Using Common Initialization Steps

Certain steps are common to all UART modes, regardless of using the core or the DMA running the transfers.

1. All UART signals are multiplexed and compete with other functions at pin level. First, the port registers need to be programmed according to the guidelines in the PORTs chapter.
2. Program the `UART_CLK` register. Refer to [UART Bit Rate Generation](#).
3. Program the `UART_CTL` register and enable the UART clock.

*RESULT:*

## Using Core Transfers

A core transmit operation is done by writing data into the `UART_THR` register, when the `UART_STAT.THRE` bit is set. If the `UART_STAT.DR` bit is set, received data can be read from the `UART_RBR` register.

## Using DMA Transfers

1. Make sure that the `UART_IMSK.ETBEI` bit or the `UART_IMSK.ERBFI` bit, respectively, are cleared before configuring the DMA.
2. Configure the dedicated DMA channel.
3. Set the `UART_IMSK.ETBEI` bit or `UART_IMSK.ERBFI` bit to start the transfer.

## Using Interrupts

Each UART features three interrupt signal outputs.

1. Enable individual interrupts in the system event controller (SEC).
2. Register IRQ handlers.
3. Use the interrupts mask registers to enable specific IRQ events.

## Setting Up Hardware Flow Control

1. Configure automatic or manual hardware flow control for the receiver through the `UART_CTL.ARTS` bit, and/or the transmitter through the `UART_CTL.ACTS` bit.
2. Configure CTS and RTS polarity through the `UART_CTL.FCPOL` bit.

## UART Register Descriptions

UART (UART) contains the following registers.

**Table 19-8:** UART Module Registers

| Name          | Description                            |
|---------------|----------------------------------------|
| UART_REVID    | Revision ID Register                   |
| UART_CTL      | Control Register                       |
| UART_STAT     | Status Register                        |
| UART_SCR      | Scratch Register                       |
| UART_CLK      | Clock Rate Register                    |
| UART_IMSK     | Interrupt Mask Register                |
| UART_IMSK_SET | Interrupt Mask Set Register            |
| UART_IMSK_CLR | Interrupt Mask Clear Register          |
| UART_RBR      | Receive Buffer Register                |
| UART_THR      | Transmit Hold Register                 |
| UART_TAIP     | Transmit Address/Insert Pulse Register |
| UART_TSR      | Transmit Shift Register                |
| UART_RSR      | Receive Shift Register                 |
| UART_TXCNT    | Transmit Counter Register              |
| UART_RXCNT    | Receive Counter Register               |

## Revision ID Register

The `UART_REVID` register provides revision information for the UART module.

**UART\_REVID: Revision ID Register - R/WE**

Reset = 0x0000 0004

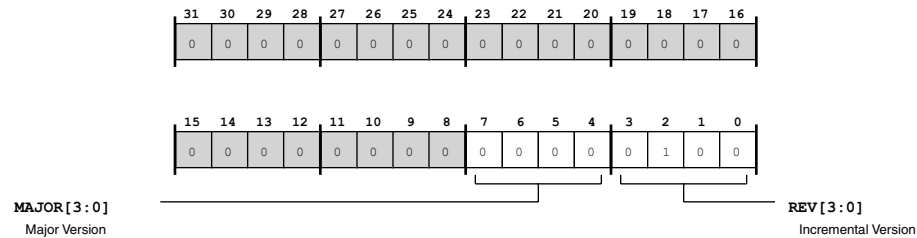


**Figure 19-14:** UART\_REVID Register Diagram

**Table 19-9:** UART\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:4     | MAJOR    | Major Version.          |
| 3:0     | REV      | Incremental Version.    |

## Control Register

The `UART_CTL` register provides enable/disable control for internal UART and for the IrDA mode of operation. This register also provides UART line control, permitting selection of the format of received and transmitted character frames. Modem feature control also is available from this register, including partial modem functionality to allow for hardware flow control and loopback mode.

UART\_CTL: Control Register - R/W

Reset = 0x0000 0000

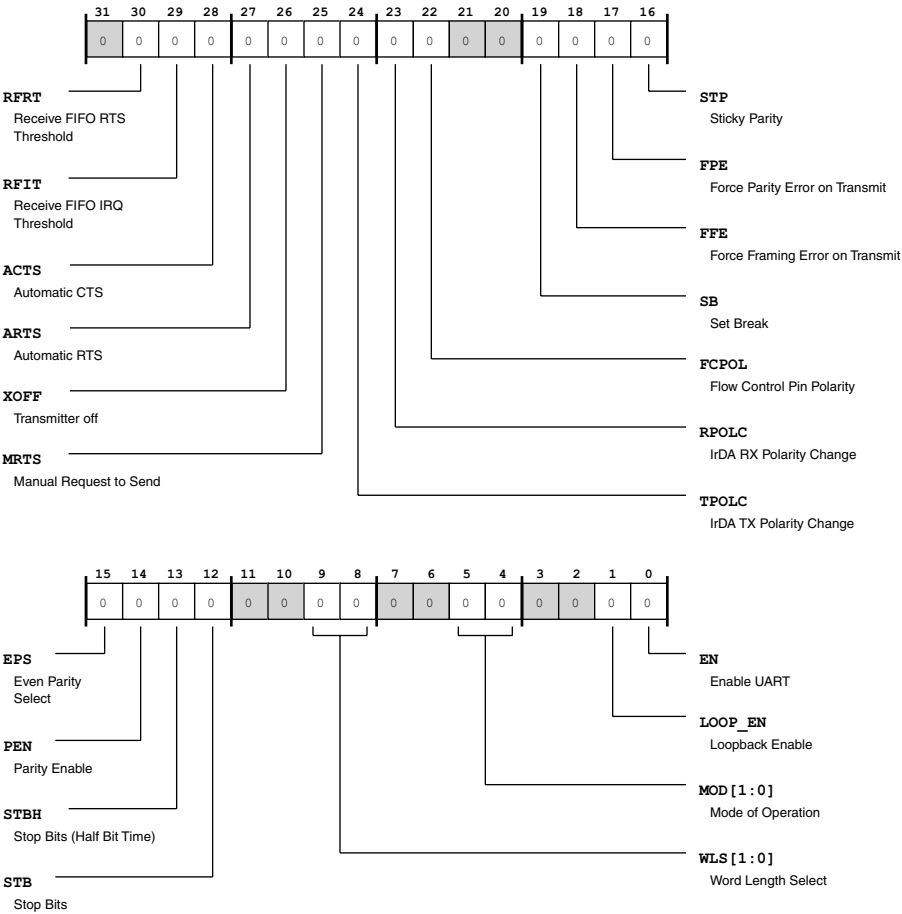


Figure 19-15: UART\_CTL Register Diagram

**Table 19-10:** UART\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30      | RFRT     | Receive FIFO RTS Threshold.<br>The <code>UART_CTL.RFRT</code> bit controls <code>UART_RTS</code> pin assertion and de-assertion timing. This bit is ignored if <code>UART_CTL.ARTS</code> is cleared. If set, the <code>UART_RTS</code> pin is de-asserted when the receive buffer already holds seven words and an eighth start bit is detected. It is re-asserted when the FIFO contains seven words or less. If cleared, de-assert <code>UART_RTS</code> pin when the RX buffer already holds four words and a fifth start bit is detected. The <code>UART_RTS</code> pin is re-asserted when the RX buffer contains no more than 4 words. |
|         |          | 0 De-assert RTS if RX FIFO word count > 4; assert if <= 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 De-assert RTS if RX FIFO word count > 7; assert if <= 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 29      | RFIT     | Receive FIFO IRQ Threshold.<br>The <code>UART_CTL.RFIT</code> bit controls the timing of the <code>UART_STAT.RFCS</code> bit. If <code>UART_CTL.RFIT</code> is cleared, the receive threshold is two. If <code>UART_CTL.RFIT</code> is set, the threshold is four words in the receive buffer.                                                                                                                                                                                                                                                                                                                                                |
|         |          | 0 Set RFCS=1 if RX FIFO count >= 4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Set RFCS=1 if RX FIFO count >= 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------|
| 28      | ACTS     | Automatic CTS.<br>The UART_CTL.ACTS bit must be set to enable the $\overline{\text{UART\_CTS}}$ input pin for $\overline{\text{UART\_TX}}$ handshaking. If enabled, the UART_STAT.CTS bit holds the value (if UART_CTL.FCPOL is set) or complement value (if UART_CTL.FCPOL is cleared) of the $\overline{\text{UART\_CTS}}$ input pin. The UART_STAT.CTS bit can be used to determine whether the external device is ready to receive data (if UART_STAT.CTS set) or whether it is busy (if UART_STAT.CTS cleared). If UART_CTL.ACTS is cleared, the $\overline{\text{UART\_TX}}$ handshaking protocol is disabled, and the $\overline{\text{UART\_TX}}$ line transmits data whenever there is data to send, regardless of the value of $\overline{\text{UART\_CTS}}$ . Software can pause ongoing transmission by setting the UART_CTL.XOFF bit. |                                  |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Disable TX handshaking protocol  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Enable TX handshaking protocol   |
| 27      | ARTS     | Automatic RTS.<br>The UART_CTL.ARTS bit must be set to enable the $\overline{\text{UART\_RTS}}$ input pin for $\overline{\text{UART\_TX}}$ handshaking.If set, hardware guarantees minimal $\overline{\text{UART\_RTS}}$ pin de-assertion pulse width of at least the number of data bits defined by the UART_CTL.WLS bit field. If cleared, the $\overline{\text{UART\_RTS}}$ pin is not generated automatically by hardware. The $\overline{\text{UART\_RTS}}$ pin can still be manually controlled by the UART_CTL.MRTS bit, and software is responsible for $\overline{\text{UART\_RTS}}$ pulse width control (if needed).                                                                                                                                                                                                                     |                                  |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Disable RX handshaking protocol. |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Enable RX handshaking protocol.  |
| 26      | XOFF     | Transmitter off.<br>The UART_CTL.XOFF bit (if set) turns off transmission (XOFF) by preventing the content of THR from being continued to TSR. When set, this bit turns on transmission (XON). The state of the UART_CTL.XOFF bit is ignored if the UART_CTL.ACTS bit is set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                  |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Transmission ON, if ACTS=0       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Transmission OFF, if ACTS=0      |



**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | MRTS     | Manual Request to Send.<br>The UART_CTL.MRTS bit controls the state of the $\overline{\text{UART\_RTS}}$ output pin when the UART_CTL.ARTS bit is cleared. When UART_CTL.MRTS is cleared, the UART de-asserts the $\overline{\text{UART\_RTS}}$ pin, signaling to the external device that the UART is not ready to receive. When UART_CTL.MRTS is set, the UART asserts the $\overline{\text{UART\_RTS}}$ pin, signaling to the external device that the UART is ready to receive. |
|         |          | 0 De-assert RTS pin when ARTS=0                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Assert RTS pin when ARTS=0                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 24      | TPOLC    | IrDA TX Polarity Change.<br>The UART_CTL.TPOLC bit selects the active low/high polarity for IrDA communications. This bit only is effective in IrDA mode. If set, in IrDA mode, the $\overline{\text{UART\_TX}}$ pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the $\overline{\text{UART\_TX}}$ pin idles low. In UART or MDB mode, it is NRZ.                                                                                                 |
|         |          | 0 Active-low TX polarity setting                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Active-high TX polarity setting                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 23      | RPOLC    | IrDA RX Polarity Change.<br>The UART_CTL.RPOLC bit selects the active low/high polarity for IrDA communications. This bit only is effective in IrDA mode. If set, in IrDA mode, the $\overline{\text{UART\_RX}}$ pin idles high. In UART or MDB mode, it is inverted-NRZ. If cleared, in IrDA mode, the $\overline{\text{UART\_RX}}$ pin idles low. In UART or MDB mode, it is NRZ.                                                                                                 |
|         |          | 0 Active-low RX polarity setting                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Active-high RX polarity setting                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | FCPOL    | Flow Control Pin Polarity.<br>The <code>UART_CTL.FCPOL</code> select the polarities of the <code>UART_CTS</code> and <code>UART_RTS</code> pins. When <code>UART_CTL.FCPOL</code> is cleared, the <code>UART_RTS</code> and <code>UART_CTS</code> pins are active low, and UART is halted when the <code>UART_RTS</code> and <code>UART_CTS</code> pin state is high. When <code>UART_CTL.FCPOL</code> is set, the <code>UART_RTS</code> and <code>UART_CTS</code> pins are active high, and UART is halted when the <code>UART_RTS</code> and <code>UART_CTS</code> pin state is low. |
|         |          | 0 Active low CTS/RTS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Active high CTS/RTS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 19      | SB       | Set Break.<br>If set, the <code>UART_CTL.SB</code> bit forces the <code>UART_TX</code> pin to low asynchronously, regardless of whether or not data is currently transmitted. This bit functions even when the UART clock is disabled. Because the <code>UART_TX</code> pin normally drives high, it can be used as a flag output pin, if the UART is not used. (For example, if <code>UART_CTL.TPOLC</code> is cleared, drive <code>UART_TX</code> pin low; or if <code>UART_CTL.TPOLC</code> is set, drive <code>UART_TX</code> pin high.)                                           |
|         |          | 0 No force                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Force TX pin to 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 18      | FFE      | Force Framing Error on Transmit.<br>The <code>UART_CTL.FFE</code> bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0 Normal operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Force error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 17      | FPE      | Force Parity Error on Transmit.<br>The <code>UART_CTL.FPE</code> bit is intended for test purposes. This bit is useful for debugging software, especially in loopback mode.                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 0 Normal operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Force parity error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| 16      | STP      | Sticky Parity.<br>The <code>UART_CTL.STP</code> bit controls whether the parity is generated by hardware based on the data bits or whether it is set to a fixed value. If this bit is cleared, the hardware calculates the parity bit value based on the data bits. Then, the EPS bit determines whether odd or even parity mode is chosen. If this bit is set, odd parity is used. That means that the total count of logical-1 data bits including the parity bit must be an odd value. Even parity is chosen by <code>UART_CTL.STP</code> cleared and <code>UART_CTL.EPS</code> set. Then, the count of logical-1 bits must be a even value. If the <code>UART_CTL.STP</code> bit is set, hardware parity calculation is disabled. In this case, the sent and received parity equals the inverted <code>UART_CTL.EPS</code> bit. |                                                                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | No Forced Parity                                               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Force (Stick) Parity to Defined Value (if <code>PEN=1</code> ) |
| 15      | EPS      | Even Parity Select.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Odd parity                                                     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Even parity                                                    |
| 14      | PEN      | Parity Enable.<br>The <code>UART_CTL.PEN</code> enables parity transmission and parity check. The <code>UART_CTL.PEN</code> bit inserts one additional bit between the most significant data bit and the first stop bit. The polarity of this so-called parity bit depends on data and the <code>UART_CTL.STP</code> and <code>UART_CTL.EPS</code> control bits. Both transmitter and receiver calculate the parity value. The receiver compares the received parity bit with the expected value and issues a parity error if they don't match. If <code>UART_CTL.PEN</code> is cleared, the <code>UART_CTL.STP</code> and the <code>UART_CTL.EPS</code> bits are ignored.                                                                                                                                                          |                                                                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Disable                                                        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Enable parity transmit and check                               |

**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | STBH     | Stop Bits (Half Bit Time).                                                                                                                                |
|         |          | 0      0 half-bit-time stop bit                                                                                                                           |
|         |          | 1      1 half-bit-time stop bit                                                                                                                           |
| 12      | STB      | Stop Bits.<br>The <code>UART_CTL.STB</code> bit controls how many stop bits are appended to transmitted data.                                             |
|         |          | 0      1 stop bit                                                                                                                                         |
|         |          | 1      2 stop bits                                                                                                                                        |
| 9:8     | WLS      | Word Length Select.<br>The <code>UART_CTL.WLS</code> field determines whether the transmitted and received UART word consists of 5, 6, 7, or 8 data bits. |
|         |          | 0      5-bit Word                                                                                                                                         |
|         |          | 1      6-bit Word                                                                                                                                         |
|         |          | 2      7-bit Word                                                                                                                                         |
|         |          | 3      8-bit Word                                                                                                                                         |
| 5:4     | MOD      | Mode of Operation.<br>The <code>UART_CTL.MOD</code> selects the UART operation mode (UMOD).                                                               |
|         |          | 0      UART Mode                                                                                                                                          |
|         |          | 1      MDB Mode                                                                                                                                           |
|         |          | 2      IrDA SIR Mode                                                                                                                                      |

**Table 19-10:** UART\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 1       | LOOP_EN  | Loopback Enable.<br>The <code>UART_CTL.LOOP_EN</code> enables UART loopback mode. When set, this bit disconnects the receiver's input from the <code>UART_RX</code> pin, and internally redirects the transmit output to the receiver. The <code>UART_TX</code> pin remains active and continues to transmit data externally as well. Loopback mode also forces the <code>UART_RTS</code> pin to its de-assertive state, disconnects the <code>UART_CTS</code> bit from the <code>UART_CTS</code> input pin, and directly connects the <code>UART_CTL.MRTS</code> bit to the <code>UART_STAT.CTS</code> bit. In loopback mode, setting the <code>UART_CTL.MRTS</code> bit sets the <code>UART_STAT.CTS</code> bit and enables the UART's transmitter. Clearing to the <code>UART_CTL.MRTS</code> bit clears the <code>UART_STAT.CTS</code> bit and disables the UART's transmitter. |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Enable  |
| 0       | EN       | Enable UART.<br>The <code>UART_CTL.EN</code> enables UART clocks. This bit also resets the state machine and control registers when cleared. Using this bit to disable the UART -- when not used -- reduces power consumption.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Enable  |

## Status Register

The `UART_STAT` register contains the UART line status and UART modem status, as indicated by the current states of the UART's `UART_CTS` pin and internal receive buffers. Writes to this register can perform write-one-to-clear (W1C) operations on most status bits. Reading this register has no side effects.

UART\_STAT: Status Register - R/W

Reset = 0x0000 00a0

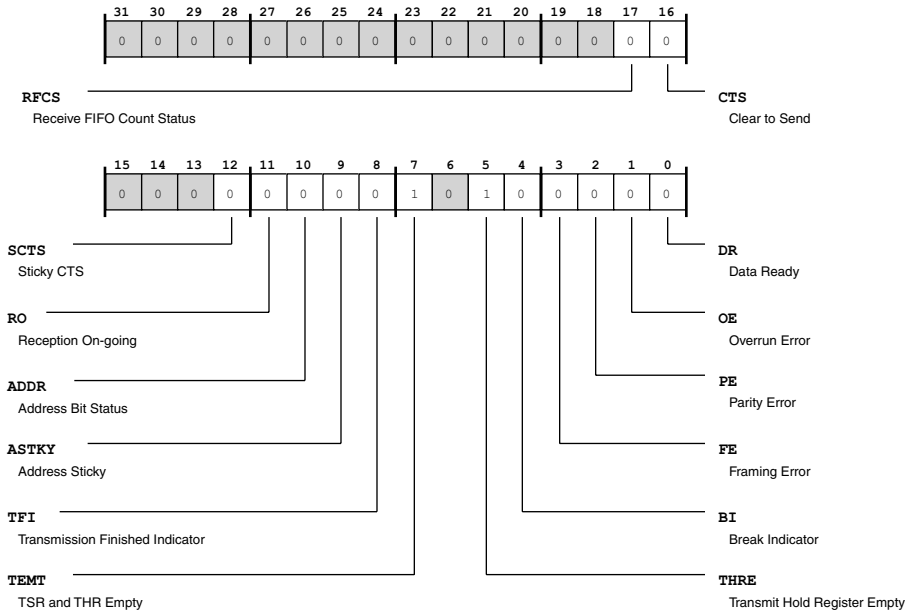


Figure 19-16: UART\_STAT Register Diagram

Table 19-11: UART\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | RFCS     | Receive FIFO Count Status.<br>The UART_STAT . RFCS bit is set when the receive buffer holds more or equal entries than a certain threshold. The threshold is controlled by the UART_CTL . RFIT bit. If UART_CTL . RFIT is cleared, the threshold is four entries. If UART_CTL . RFIT is set, the threshold is seven entries. The UART_STAT . RFCS bit is cleared when the UART_RBR register is read sufficient times until the buffer is drained below the threshold. The UART_STAT . RFCS bit can trigger a status interrupt if enabled by the UART_IMSK_SET . ERFCI bit. |
|         |          | 0 RX FIFO has less than 4 (7) entries when RFIT=0 (1)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 RX FIFO has at least 4 (7) entries when RFIT=0 (1)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Table 19-11:** UART\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                          |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| 16      | CTS      | Clear to Send.<br>The <code>UART_STAT.CTS</code> bit holds the value (if <code>UART_CTL.FCPOL</code> set) or the complement value (if <code>UART_CTL.FCPOL</code> cleared) of the <code>UART_CTS</code> input pin. The <code>UART_CTL.ACTS</code> bit must be set to enable this feature. The core can read the value of the <code>UART_STAT.CTS</code> bit to determine whether the external device is ready to receive ( <code>UART_STAT.CTS</code> set) or if it is busy ( <code>UART_STAT.CTS</code> cleared). If <code>UART_CTL.ACTS</code> is cleared, the <code>UART_TX</code> handshaking protocol is disabled, and the UART transmits data as long as there is data to transmit, regardless of the value of <code>UART_STAT.CTS</code> . When <code>UART_CTL.ACTS</code> is cleared, the software can pause transmission temporarily by setting the <code>XOFF</code> bit. Note that in loopback mode ( <code>UART_CTL.LOOP_EN</code> set), the <code>UART_STAT.CTS</code> bit is disconnected from the <code>UART_CTS</code> input pin. Instead, the bit is directly connected to the <code>UART_CTL.MRTS</code> bit. |                                                          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Not clear to send (External device not ready to receive) |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Clear to send (External device ready to receive)         |
| 12      | SCTS     | Sticky CTS.<br>The <code>UART_STAT.SCTS</code> bit is a sticky bit that is set when <code>UART_STAT.CTS</code> transitions from 0 to 1. The <code>UART_STAT.SCTS</code> bit is cleared by software with a <code>W1C</code> operation. This bit can trigger a line status interrupt if enabled by the <code>UART_IMSK_SET.EDSSI</code> bit.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | CTS has not transitioned from low to high                |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | CTS has transitioned from low to high                    |
| 11      | RO       | Reception On-going.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | No data reception in progress                            |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Data reception in progress                               |

**Table 19-11:** UART\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | ADDR     | Address Bit Status.<br>The <code>UART_STAT.ADDR</code> bit is used to mirror the address bit of the word in <code>UART_RBR</code> in multi-drop bus protocol, and is enabled only in MDB mode. The <code>UART_STAT.ADDR</code> bit is updated by hardware upon detecting a received word with the address bit in <code>UART_RBR</code> set or cleared. Additionally, software can set the ADDR bit with a write-1-to-set (W1S) operation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 0 Address bit is low                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 Address bit is high                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 9       | ASTKY    | Address Sticky.<br>The <code>UART_STAT.ASTKY</code> bit is used in multi-drop bus mode to indicate whether a peripheral is currently being addressed. This bit is a sticky version of the <code>UART_STAT.ADDR</code> bit and is set by hardware when setting the <code>UART_STAT.ADDR</code> bit. The <code>UART_STAT.ASTKY</code> bit can only be cleared by software with a write-one-to-clear (W1C) operation. With the <code>UART_STAT.ASTKY</code> bit set, words will be received irrespective of the <code>UART_CTL.MOD</code> bit or <code>UART_STAT.ADDR</code> bit selection. With the <code>UART_STAT.ASTKY</code> bit cleared, only address words ( <code>UART_CTL.MOD</code> bit set) will be received and words with <code>UART_CTL.MOD</code> bit cleared are ignored (not moved from the RSR to the RX FIFO) in MDB mode. The <code>UART_STAT.ASTKY</code> bit does not affect reception in non-MDB modes. |
|         |          | 0 ADDR bit has not been set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 ADDR bit has been set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 8       | TFI      | Transmission Finished Indicator.<br>The <code>UART_STAT.TFI</code> bit is a sticky version of the <code>UART_STAT.TEMT</code> bit. While <code>UART_STAT.TEMT</code> is automatically cleared by hardware when new data is written to the <code>UART_THR</code> register, the sticky <code>UART_STAT.TFI</code> bit remains set, until it is cleared by software (W1C). The <code>UART_STAT.TFI</code> bit enables more flexible transmit interrupt timing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 0 TEMT did not transition from 0 to 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 TEMT transition from 0 to 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |



**Table 19-11:** UART\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | TEMT     | TSR and THR Empty.<br>The UART_STAT.TEMT bit indicates that the UART_THR and UART_TAIP registers and the UART_TSR register are empty. In this case, the program is permitted to write to the UART_THR and UART_TAIP registers twice without losing data. The UART_STAT.TEMT bit can also be used as indicator that pending UART transmission is completed. At that time, it is safe to disable the UART_CTL.EN bit or to three-state the off-chip line driver.                                                                |
|         |          | 0 Not empty TSR/THR                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 TSR/THR Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 5       | THRE     | Transmit Hold Register Empty.<br>The UART_STAT.THRE bit indicates that the UART transmit channel is ready for new data and software can write to the UART_THR and UART_TAIP registers. Writes to the UART_THR and UART_TAIP registers clear the UART_STAT.THRE. The bit is set again when the UART_THR and UART_TAIP registers are empty and ready to accept data.                                                                                                                                                            |
|         |          | 0 Not empty THR/TAIP                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Empty THR/TAIP                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 4       | BI       | Break Indicator.<br>The UART_STAT.BI bit indicates that the first stop bit is sampled low and the entire data word, including parity bit, consists of low bits only. (This condition indicates that UART_RX was held low for more than the maximum word length.) The UART_STAT.BI bit is updated simultaneously with the UART_STAT.DR bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the UART_RBR register. The bit is sticky and can be cleared by W1C operations. |
|         |          | 0 No break interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Break interrupt<br>this indicates UARTxRX was held low(RPOLC=0) / high (RPOLC=1) for more than the maximum word length                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 19-11:** UART\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| 3       | FE       | Framing Error.<br>The <code>UART_STAT.FE</code> bit indicates that the first stop bit is sampled. This bit is updated simultaneously with the <code>UART_STAT.DR</code> bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the <code>UART_RBR</code> register. The <code>UART_STAT.FE</code> bit is sticky and can be cleared by W1C operations. Note that invalid stop bits can be simulated by setting the <code>UART_CTL.FFE</code> bit.                                                                                                                                                                                                              |                        |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No error               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Invalid stop bit error |
| 2       | PE       | Parity Error.<br>The <code>UART_STAT.PE</code> bit indicates that the received parity bit does not match the expected value. This bit is updated simultaneously with the <code>UART_STAT.DR</code> bit, that is, by the time the first stop bit is received or when data is loaded from the receive FIFO to the <code>UART_RBR</code> register. The <code>UART_STAT.PE</code> bit is sticky and can be cleared by W1C operations. Note that invalid parity bits can be simulated by setting the <code>UART_CTL.FPE</code> bit.                                                                                                                                                                                 |                        |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No parity error        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Parity error           |
| 1       | OE       | Overrun Error.<br>The <code>UART_STAT.OE</code> bit indicates that further data is received while the internal receive buffer was full. This bit is set when sampling the stop bit of the sixth data word. To avoid overruns, read the <code>UART_RBR</code> register in time. In DMA receive mode, overruns are very unlikely to happen ever. After an overrun occurs, the <code>UART_RBR</code> and receive FIFO are protected from being overwritten by new data until the <code>UART_STAT.OE</code> bit is cleared by software. The content of the <code>UART_RSR</code> register is lost as soon as the overrun occurs. The <code>UART_STAT.OE</code> bit is sticky and can be cleared by W1C operations. |                        |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No overrun             |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Overrun error          |

**Table 19-11:** UART\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | DR       | Data Ready.<br>The UART_STAT.DR bit indicates that data is available in the receiver and can be read from the UART_RBR register. The bit is set by hardware when the receiver detects the first valid stop bit. The bit is cleared by hardware when the UART_RBR register is read. |
|         |          | 0 No new data                                                                                                                                                                                                                                                                      |
|         |          | 1 New data in RBR                                                                                                                                                                                                                                                                  |

## Scratch Register

The UART\_SCR registers contain 8-bit scratch pad data. These registers are used for general purpose data storage and do not control the UART hardware in any way.

### UART\_SCR: Scratch Register - R/W

Reset = 0x0000 0000

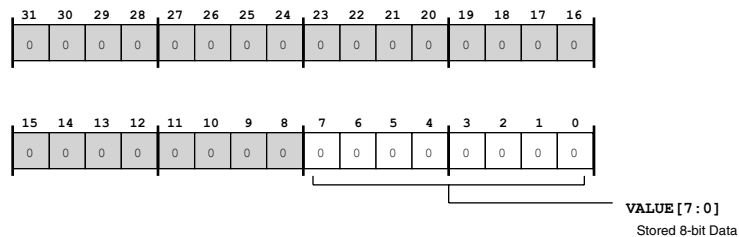


**Figure 19-17:** UART\_SCR Register Diagram

**Table 19-12:** UART\_SCR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | Stored 8-bit Data.      |

## Clock Rate Register

The UART\_CLK register divides the system clock (SCLK) down to the bit clock.

UART\_CLK: Clock Rate Register - R/W

Reset = 0x0000 ffff

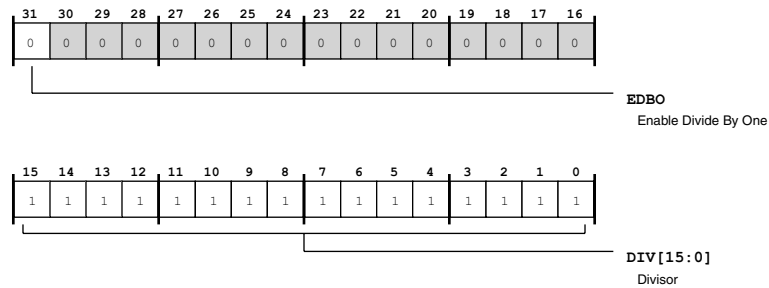


Figure 19-18: UART\_CLK Register Diagram

Table 19-13: UART\_CLK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | EDBO     | Enable Divide By One.<br>The UART_CLK.EDBO bit enables bypassing of the divide-by-16 prescaler in bit clock generation. This improves bit rate granularity, especially at high bit rates. Do not set this bit in IrDA mode. |
|         |          | 0 Bit clock prescaler = 16                                                                                                                                                                                                  |
|         |          | 1 Bit clock prescaler = 1                                                                                                                                                                                                   |
| 15:0    | DIV      | Divisor.<br>The UART_CLK.DIV provides the divisor for the UART's clock bit rate calculation.<br>The bit rate is defined by:<br>Bit Rate = SCLK / (16 <sup>(1-EDBo)</sup> x UART_CLK.DIV)                                    |

Interrupt Mask Register

The UART\_IMSK indicates interrupt mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART\_IMSK\_SET and UART\_IMSK\_CLR register pair. Writing ones to UART\_IMSK\_SET enables (unmasks) interrupts, and writing ones to UART\_IMSK\_CLR disables (masks) them. Reads from either register return the enabled bits.

The UART\_IMSK register is used to enable requests for system handling of empty or full states of UART data registers. Unless polling is used as a means of action, the UART\_IMSK.ERBFI and/or UART\_IMSK.ETBEI bits are normally set. Setting this register without enabling system DMA causes the UART to notify the processor of data inventory state by means of interrupts. For proper operation in this mode, system interrupts must be enabled, and appropriate interrupt handling routines must be present.

Each UART features three separate interrupt channels to handle data transmit, data receive, and line status events independently, regardless whether DMA is enabled or not. If no DMA channels are assigned to the UART, set the `UART_IMSK.ELSI` bit to reroute transmit and receive interrupts to the status interrupt output.

With system DMA enabled, the UART uses DMA to transfer data to or from the processor. Dedicated DMA channels are available to receive and transmit operation. Line error handling can be configured completely independently from the receive/transmit setup.

The UART's DMA is enabled by first setting up the system DMA control registers and then enabling the `UART_IMSK.ERBFI` and/or `UART_IMSK.ETBEI` interrupts. This is because the interrupt request lines double as DMA request lines. Depending on whether DMA is enabled or not, upon receiving these requests, the DMA control unit either generates a direct memory access or passes the UART interrupt on to the system interrupt handling unit. However, UART's error interrupt goes directly to the system interrupt handling unit, bypassing the DMA unit completely.

#### UART\_IMSK: Interrupt Mask Register - R/W

Reset = 0x0000 0000

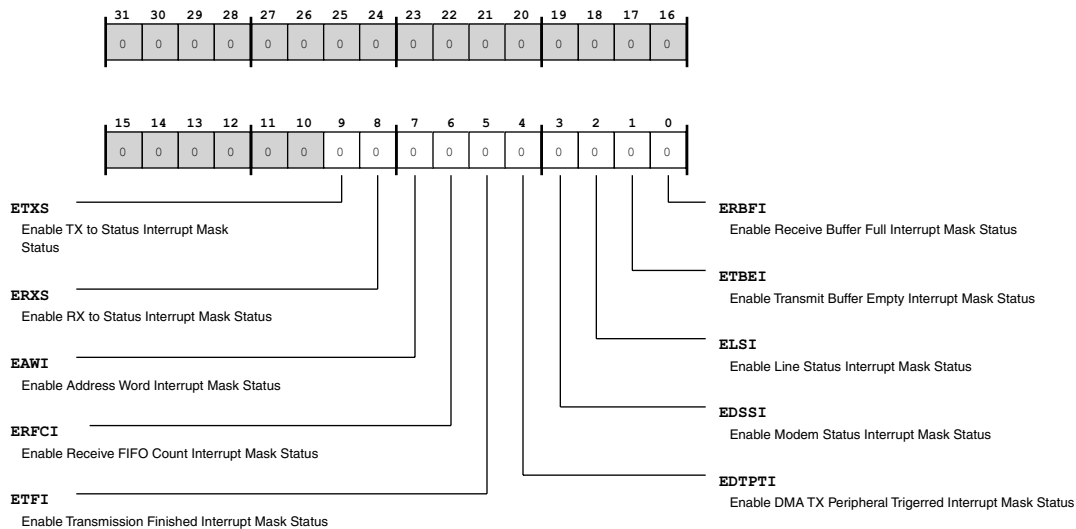


**Figure 19-19:** UART\_IMSK Register Diagram

**Table 19-14:** UART\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | ETXS     | Enable TX to Status Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ETXS</code> bit indicates re-direction of the TX interrupts to status interrupt output. If cleared, TX interrupts are routed to normal interrupt outputs.                                                                        |
|         |          | 0                      Interrupt is masked                                                                                                                                                                                                                                                                                      |
|         |          | 1                      Interrupt is unmasked                                                                                                                                                                                                                                                                                    |
| 8       | ERXS     | Enable RX to Status Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ERXS</code> bit indicates re-direction of RX interrupts to status interrupt output. If cleared, RX interrupts are routed to normal interrupt outputs.                                                                            |
|         |          | 0                      Interrupt is masked                                                                                                                                                                                                                                                                                      |
|         |          | 1                      Interrupt is unmasked                                                                                                                                                                                                                                                                                    |
| 7       | EAWI     | Enable Address Word Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.EAWI</code> bit indicates generation of a status interrupt when an Address word in MDB-mode is present in the <code>UART_RBR</code> . A received word is an address word if the <code>UART_STAT.ADDR</code> bit is set.          |
|         |          | 0                      Interrupt is masked                                                                                                                                                                                                                                                                                      |
|         |          | 1                      Interrupt is unmasked                                                                                                                                                                                                                                                                                    |
| 6       | ERFCI    | Enable Receive FIFO Count Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ERFCI</code> bit indicates enabling of the receive buffer threshold interrupt if signaled by the <code>UART_STAT.RFCS</code> bit. Read the <code>UART_RBR</code> register sufficient times to clear the interrupt request. |
|         |          | 0                      Interrupt is masked                                                                                                                                                                                                                                                                                      |
|         |          | 1                      Interrupt is unmasked                                                                                                                                                                                                                                                                                    |

**Table 19-14:** UART\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | ETFI     | Enable Transmission Finished Interrupt Mask Status.<br>If set (interrupt unmasked) the <code>UART_IMSK.ETFI</code> bit indicates enabling of interrupt generation on the status interrupt channel when the transmit buffer register, the transmit address register, and the transmit shift register are all empty as indicated by the <code>UART_STAT.TFI</code> . The <code>UART_IMSK.ETFI</code> interrupt can be used to avoid expensive polling of the <code>UART_STAT.TEMT</code> bit, when the UART clock or line drivers should be disabled after transmission has completed. W1C the <code>UART_STAT.TFI</code> bit to clear the interrupt request. In DMA operation, the <code>UART_IMSK.ETFI</code> bit's functionality might be preferred.                                                                                                                                                            |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 4       | EDTPTI   | Enable DMA TX Peripheral Triggerred Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.EDTPTI</code> bit indicates enabling of the DMA completion interrupt to be delayed until the data has left the UART completely. This bit is required for DMA transmit operation only. If set, the UART can generate a DMA interrupt by the time the <code>UART_STAT.TEMT</code> bit goes high after the last DMA data word is transmitted.<br>When <code>UART_IMSK.EDTPTI</code> is set, usually the <code>DDE_CFG_INT</code> field is cleared to "00" in a STOP mode DMA. This set up suppresses the normal completion interrupt, and the <code>UART_STAT.TEMT</code> event is signaled through the DMA controller and triggers the DMA interrupt. If both ( <code>DDE_CFG_INT</code> not 00 and <code>UART_IMSK.EDTPTI</code> set), two interrupts are requested at the end of a STOP mode DMA. |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

**Table 19-14:** UART\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | EDSSI    | Enable Modem Status Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.EDSSI</code> bit indicates enabling of a modem status interrupt on the same status interrupt channel when the <code>UART_STAT.SCTS</code> bit is set. This indicates <code>UART_CTS</code> pin re-assertion. Write-1-to-clear (W1C) the <code>UART_STAT.SCTS</code> bit to clear the interrupt request.                                                                                                                                                                        |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 2       | ELSI     | Enable Line Status Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ELSI</code> bit indicates that redirection of TX and RX interrupt requests to the status interrupt output of the UART by OR'ing them with the <code>UART_STAT.OE</code> , <code>UART_STAT.PE</code> , <code>UART_STAT.FE</code> , and <code>UART_STAT.BI</code> interrupt requests. Set this bit when no DMA channel is associated with the UART. Enabling <code>UART_IMSK.ELSI</code> disables the RX/TX interrupt channels and negates the <code>UART_IMSK.EDTPTI</code> bit. |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 1       | ETBEI    | Enable Transmit Buffer Empty Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ETBEI</code> bit indicates generation of a TX interrupt if the <code>UART_STAT.THRE</code> bit is set.                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 0       | ERBFI    | Enable Receive Buffer Full Interrupt Mask Status.<br>If set (interrupt unmasked), the <code>UART_IMSK.ERBFI</code> indicates generation of an RX interrupt if the <code>UART_STAT.DR</code> bit is set.                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 0 Interrupt is masked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Interrupt is unmasked                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |



## Interrupt Mask Set Register

The UART\_IMSK indicates interrupt mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART\_IMSK\_SET and UART\_IMSK\_CLR register pair. Writing ones to UART\_IMSK\_SET enables (unmasks) interrupts, and writing ones to UART\_IMSK\_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART\_IMSK register description.

### UART\_IMSK\_SET: Interrupt Mask Set Register - R/W

Reset = 0x0000 0000

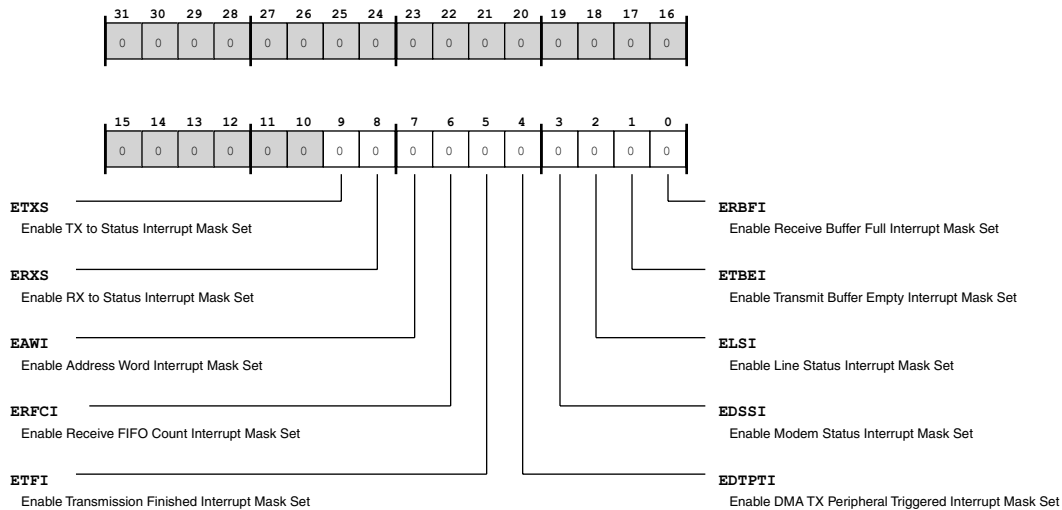


Figure 19-20: UART\_IMSK\_SET Register Diagram

Table 19-15: UART\_IMSK\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                 |
|---------|----------|-----------------------------------------|
| 9       | ETXS     | Enable TX to Status Interrupt Mask Set. |
|         |          | 0 No action                             |
|         |          | 1 Unmask interrupt                      |
| 8       | ERXS     | Enable RX to Status Interrupt Mask Set. |
|         |          | 0 No action                             |
|         |          | 1 Unmask interrupt                      |

**Table 19-15:** UART\_IMSK\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                |                  |
|---------|----------|--------------------------------------------------------|------------------|
| 7       | EAWI     | Enable Address Word Interrupt Mask Set.                |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 6       | ERFCI    | Enable Receive FIFO Count Interrupt Mask Set.          |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 5       | ETFI     | Enable Transmission Finished Interrupt Mask Set.       |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 4       | EDTPTI   | Enable DMA TX Peripheral Triggered Interrupt Mask Set. |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 3       | EDSSI    | Enable Modem Status Interrupt Mask Set.                |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 2       | ELSI     | Enable Line Status Interrupt Mask Set.                 |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |
| 1       | ETBEI    | Enable Transmit Buffer Empty Interrupt Mask Set.       |                  |
|         |          | 0                                                      | No action        |
|         |          | 1                                                      | Unmask interrupt |

**Table 19-15:** UART\_IMSK\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                        |
|---------|----------|------------------------------------------------|
| 0       | ERBFI    | Enable Receive Buffer Full Interrupt Mask Set. |
|         |          | 0 No action                                    |
|         |          | 1 Unmask interrupt                             |

## Interrupt Mask Clear Register

The UART\_IMSK indicates interrupt mask status (unmasked if set, masked if cleared) of UART status interrupts. This register is not a data register. Instead it is controlled by the UART\_IMSK\_SET and UART\_IMSK\_CLR register pair. Writing ones to UART\_IMSK\_SET enables (unmasks) interrupts, and writing ones to UART\_IMSK\_CLR disables (masks) them. Reads from either register return the enabled bits. For more information, see the UART\_IMSK register description.

### UART\_IMSK\_CLR: Interrupt Mask Clear Register - R/W

Reset = 0x0000 0000

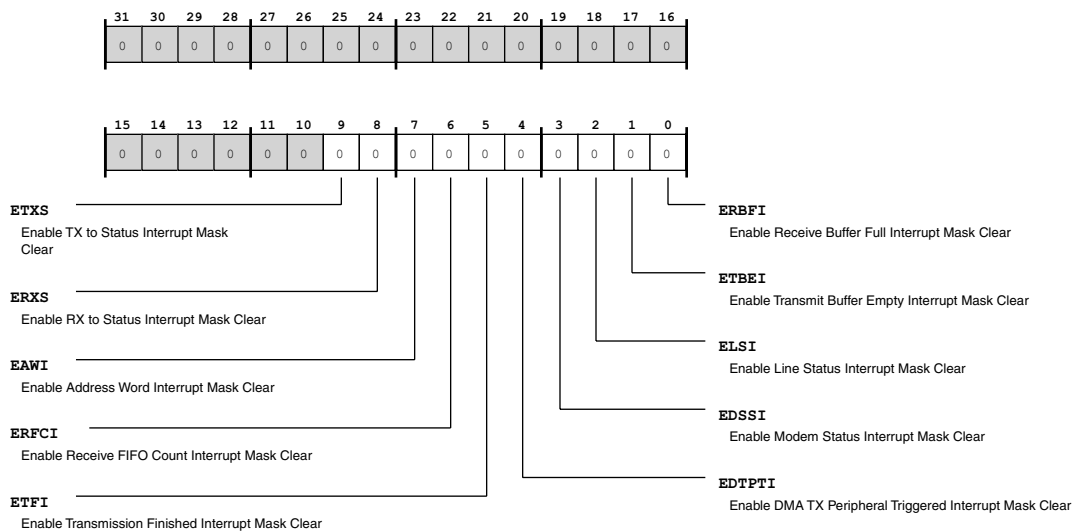


**Figure 19-21:** UART\_IMSK\_CLR Register Diagram

**Table 19-16:** UART\_IMSK\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                  |                |
|---------|----------|----------------------------------------------------------|----------------|
| 9       | ETXS     | Enable TX to Status Interrupt Mask Clear.                |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 8       | ERXS     | Enable RX to Status Interrupt Mask Clear.                |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 7       | EAWI     | Enable Address Word Interrupt Mask Clear.                |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 6       | ERFCI    | Enable Receive FIFO Count Interrupt Mask Clear.          |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 5       | ETFI     | Enable Transmission Finished Interrupt Mask Clear.       |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 4       | EDTPTI   | Enable DMA TX Peripheral Triggered Interrupt Mask Clear. |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |
| 3       | EDSSI    | Enable Modem Status Interrupt Mask Clear.                |                |
|         |          | 0                                                        | No action      |
|         |          | 1                                                        | Mask interrupt |

**Table 19-16:** UART\_IMSK\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 2       | ELSI     | Enable Line Status Interrupt Mask Clear.           |
|         |          | 0 No action                                        |
|         |          | 1 Mask interrupt                                   |
| 1       | ETBEI    | Enable Transmit Buffer Empty Interrupt Mask Clear. |
|         |          | 0 No action                                        |
|         |          | 1 Mask interrupt                                   |
| 0       | ERBFI    | Enable Receive Buffer Full Interrupt Mask Clear.   |
|         |          | 0 No action                                        |
|         |          | 1 Mask interrupt                                   |

## Receive Buffer Register

The read-only `UART_RBR` register is the UART's receive buffer. It is updated when there is pending data in the receive FIFO. Newly available data is signaled by the `UART_STAT.DR` bit.

### UART\_RBR: Receive Buffer Register - R/WE

Reset = 0x0000 0000

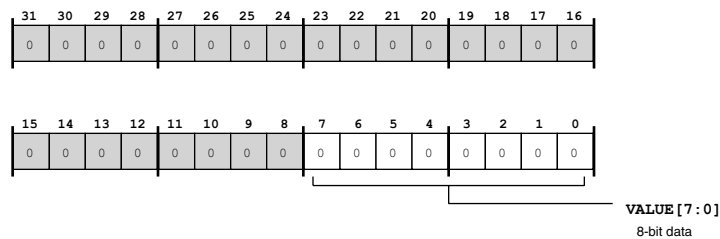


**Figure 19-22:** UART\_RBR Register Diagram

**Table 19-17:** UART\_RBR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | 8-bit data.             |

## Transmit Hold Register

The write-only `UART_THR` register is the UART's transmit buffer. The `UART_STAT.THRE` bit indicates whether data can be written to `UART_THR`. Writes to this register automatically propagate to the internal `UART_TSR` register as soon as `UART_TSR` is ready. Then, transmit operation is initiated immediately.

**UART\_THR: Transmit Hold Register - R/W**

Reset = 0x0000 0000

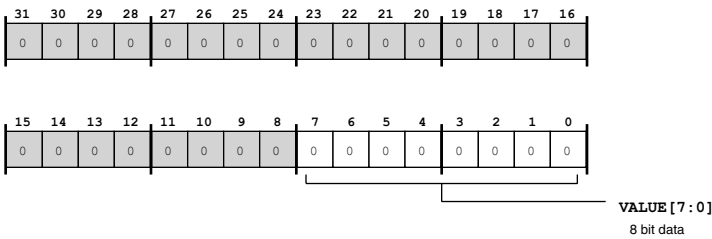


**Figure 19-23:** UART\_THR Register Diagram

**Table 19-18:** UART\_THR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | 8 bit data.             |

## Transmit Address/Insert Pulse Register

The `UART_TAIP` register and the `UART_THR` register share the same physical register, but `UART_TAIP` has different effect than the `UART_THR` register when `UART_TAIP` is written to in MDB and UART modes.

In MDB mode, data written to the `UART_TAIP` register is transmitted as an address frame (as with the `UART_CTL.MOD` bit set).

In UART mode, a write to `UART_TAIP` causes a pulse of value `UART_TAIP` [7] for a duration of `UART_TAIP` [6:0] x bit time. (There is additional inversion if the `UART_CTL.TPOLC` bit is set).

Bit time is defined by the `UART_CLK` register. The transmission of the pulse is followed by stop bit transmission as specified by the `UART_CTL.STB` and `UART_CTL.STBH` bits. This could be used for supporting line break command and inter-frame gap.

In IrDA mode, writes to `UART_TAIP` is treated the same as writes to `UART_THR`.

Accesses to the `UART_TAIP` register have the same affects as the `UART_THR` register with respect to the `UART_STAT.THRE`, `UART_STAT.TEMT`, and `UART_STAT.TFI` flags.

**UART\_TAIP: Transmit Address/Insert Pulse Register - R/W**

Reset = 0x0000 0000

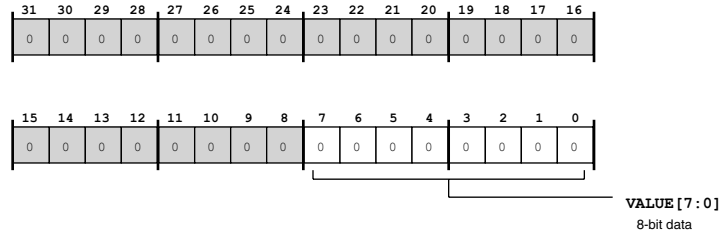


**Figure 19-24:** UART\_TAIP Register Diagram

**Table 19-19:** UART\_TAIP Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | 8-bit data.             |

## Transmit Shift Register

The read only `UART_TSR` register which returns the content of the UART's transmit shift register.

**UART\_TSR: Transmit Shift Register - R/WE**

Reset = 0x0000 07ff

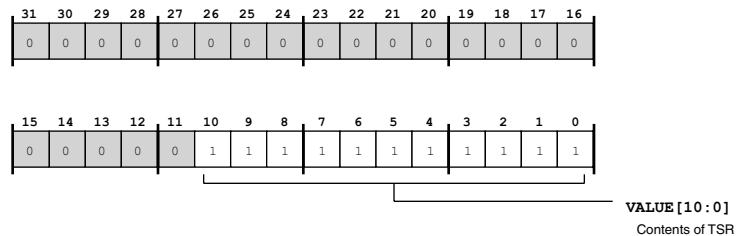


**Figure 19-25:** UART\_TSR Register Diagram

**Table 19-20:** UART\_TSR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 10:0    | VALUE    | Contents of TSR.        |

## Receive Shift Register

The read only `UART_RSR` register which returns the content of the UART's receive shift register.

The frame data is moved into this shift register after polarity inversion, if any (including the native polarity inversion in the IrDA case).

In the case of the longest frame (MDB, with parity mode, and 8 bit data word-length), the start bit may be shifted out and not available for reading at the end of the frame reception. This register is NOT reset at the start of frame. If read, in the middle of a frame reception, data corresponding the previous frame may not have entirely shifted out (for example, the read data that have been read may NOT correspond entirely to the frame being received).

Because the UART is receiving only 1 stop bit, the `UART_RSR` contains only 1 stop bit even if more than one stop bit is present in the actual transfer. This register may be considered as storing the 10 most recently received bits (taking into consideration the stop bit receive limitation above).

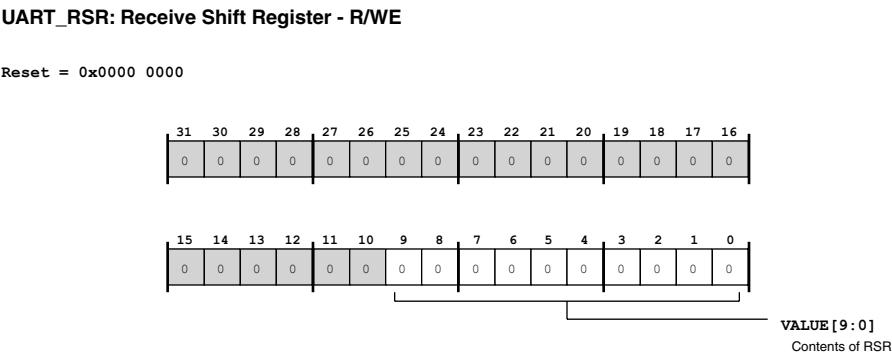


Figure 19-26: UART\_RSR Register Diagram

Table 19-21: UART\_RSR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 9:0     | VALUE    | Contents of RSR.        |

Transmit Counter Register

The `UART_TXCNT` read only register returns the content of 16-bit counter in the UART transmitter. This count is used for baud rate clock generation (the lower [15:0] is the count data).



**UART\_TXCNT: Transmit Counter Register - R/WE**

Reset = 0x0000 0000

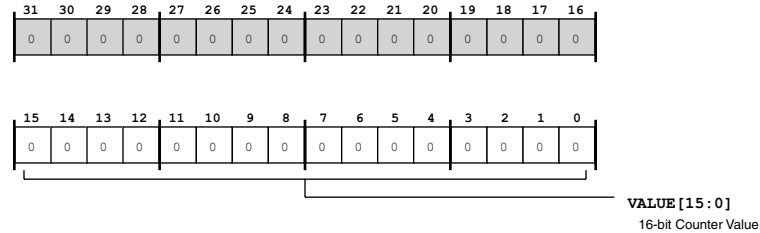


**Figure 19-27:** UART\_TXCNT Register Diagram

**Table 19-22:** UART\_TXCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | 16-bit Counter Value.   |

## Receive Counter Register

The `UART_RXCNT` register returns the content of 16-bit counter in the UART receiver. This count is used for baud rate clock generation (the lower [15:0] is the count data).

**UART\_RXCNT: Receive Counter Register - R/WE**

Reset = 0x0000 0000

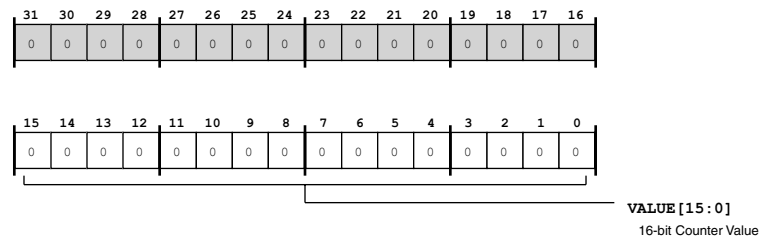


**Figure 19-28:** UART\_RXCNT Register Diagram

**Table 19-23:** UART\_RXCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | 16-bit Counter Value.   |



## 20 2-Wire Interface (TWI)

The processors include a 2-wire interface (TWI), providing a simple exchange method of control data between multiple devices. The TWI module is compatible with the widely used I<sup>2</sup>C bus standard. The TWI module offers the capabilities of simultaneous master and slave operation and support for both 7-bit addressing and multimedia data arbitration. The TWI interface utilizes two pins for transferring clock (TWI\_SCL) and data (TWI\_SDA) and supports the protocol at speeds up to 400k bits/sec. The TWI interface pins are compatible with 5 V logic levels.

Additionally, the TWI module is fully compatible with serial camera control bus (SCCB) functionality for easier control of various CMOS camera sensor devices.

To preserve processor bandwidth, the TWI module can be set up with transfer initiated interrupts only to service FIFO buffer data reads and writes. Protocol related interrupts are optional. The TWI externally moves 8-bit data while maintaining compliance with the I<sup>2</sup>C bus protocol.

### TWI Features

The TWI is fully compatible with the widely used I<sup>2</sup>C bus standard.

The TWI controller includes these features.

- Simultaneous master and slave operation on multiple device systems
- Support for multi-master bus arbitration
- 7-bit addressing
- 100K bits/second and 400K bits/second data rates
- General call address support
- Master clock synchronization and support for clock low extension
- Separate multiple-byte receive and transmit FIFOs
- Low interrupt rate
- Individual override control of data and clock lines in the event of bus lock-up
- Input filter for spike suppression
- Serial camera control bus support as specified in the *OmniVision Serial Camera Control Bus (SCCB) Functional Specification*

## TWI Functional Description

The TWI interface is a shift register that serially transmits and receives data bits, one bit at a time at the SCL rate, to and from other TWI devices. The SCL signal synchronizes the shifting and sampling of the data on the serial data pin.

## TWI Register List

The 2-wire interface TWI controller allows a device to interface to an inter IC bus as specified by the Philips I<sup>2</sup>C Bus Specification version 2.1 dated January 2000. A set of registers govern TWI operations. For more information on TWI functionality, see the TWI register descriptions.

**Table 20-1:** TWI Module Registers

| Name         | Description                   |
|--------------|-------------------------------|
| TWI_CLKDIV   | SCL Clock Divider Register    |
| TWI_CTL      | Control Register              |
| TWI_SLVCTL   | Slave Mode Control Register   |
| TWI_SLVSTAT  | Slave Mode Status Register    |
| TWI_SLVADDR  | Slave Mode Address Register   |
| TWI_MSTRCTL  | Master Mode Control Registers |
| TWI_MSTRSTAT | Master Mode Status Register   |
| TWI_MSTRADDR | Master Mode Address Register  |
| TWI_ISTAT    | Interrupt Status Register     |
| TWI_IMSK     | Interrupt Mask Register       |
| TWI_FIFOCTL  | FIFO Control Register         |
| TWI_FIFOSTAT | FIFO Status Register          |
| TWI_TXDATA8  | Tx Data Single-Byte Register  |
| TWI_TXDATA16 | Tx Data Double-Byte Register  |
| TWI_RXDATA8  | Rx Data Single-Byte Register  |

**Table 20-1:** TWI Module Registers (Continued)

| Name         | Description                  |
|--------------|------------------------------|
| TWI_RXDATA16 | Rx Data Double-Byte Register |

## ADSP-BF60x TWI Interrupt List

**Table 20-2:** ADSP-BF60x TWI Interrupt List

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| TWI0 Data Interrupt | 32 |             | LEVEL       |
| TWI1 Data Interrupt | 33 |             | LEVEL       |

## TWI Block Diagram

The following figure shows the basic blocks of the TWI interface.

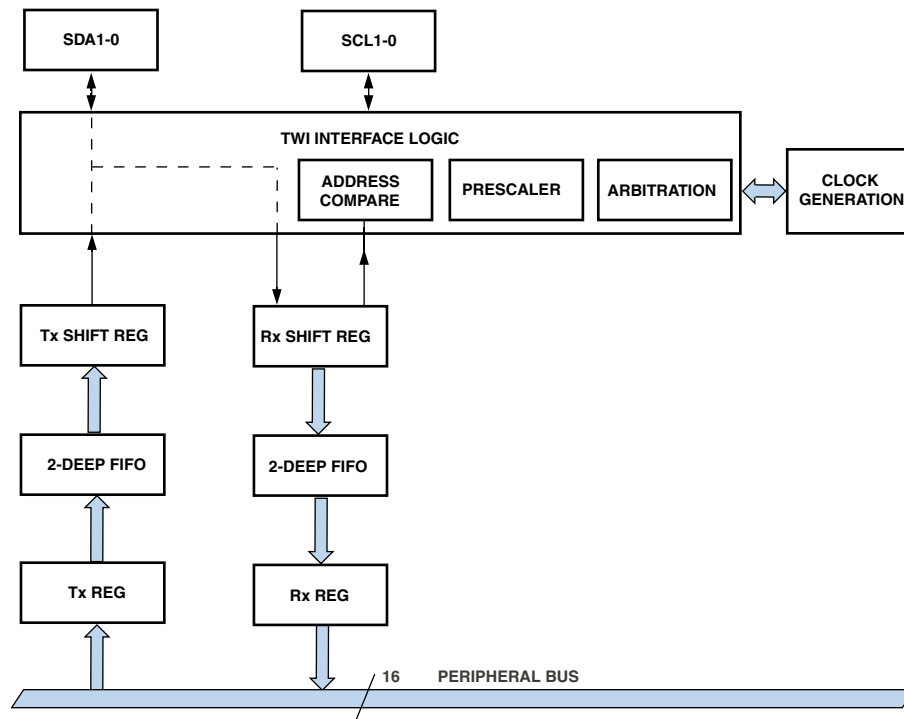


**Figure 20-1:** TWI Block Diagram

## TWI External Interface

The SDA (serial data) and SCL (serial clock) signals are open drain and require pull-up resistors. These bidirectional signals externally interface the TWI controller to the I<sup>2</sup>C bus. The interface is simple and no other external connections or logic are required.

### Serial Clock Signal (SCL)

The serial clock signal (SCL) is an input in slave mode. In master mode the TWI controller must set this signal to the desired frequency.

The TWI controller supports the standard mode of operation (up to 100 kHz) or fast mode (up to 400 kHz). The TWI control register (CTL) is used to set the PRESCALE value which sets the relationship between the system clock (SCLK) and the TWI controller's internally timed events. The internal time reference is derived from SCLK using a prescaled value. The PRESCALE value is the number of SCLK periods used in the generation of one internal time reference. The value of PRESCALE must be set to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value as shown below.

$$\text{PRESCALE} = f_{\text{SCLK}}/10\text{MHz}$$

During master mode operation, the CLKDIV register values are used to create the minimum CLKHI and CLKLO durations of the serial clock (SCL). The CLKHI field of the CLKDIV register specifies the minimum number of 10 MHz time reference periods the serial clock (SCL) waits before a new clock low period begins, assuming a single master. It is represented as an 8-bit binary value. The CLKLO field of the CLKDIV register specifies the minimum number of internal time reference periods the serial clock (SCL) is held low. It is represented as an 8-bit binary value.

Serial clock frequencies can vary from 400 kHz to less than 20 kHz. The resolution of the clock generated is 1/10 MHz or 100 ns as shown below.

$$\text{CLKDIV} = \text{TWI SCL period} / 10 \text{ MHz time reference.}$$

For example, for an SCL of 400 kHz (period = 1/400 kHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns) the following equation is used:

$$\text{CLKDIV} = 2500 \text{ ns} / 100 \text{ ns} = 25$$

Therefore, a SCL with a 30% duty cycle has CLKLO = 17 and CLKHI = 8. Note that CLKLO and CLKHI add up to CLKDIV.

**NOTE:** The CLKHI and CLKLO fields of the CLKDIV register are not intended to guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for SCL. Falling edges are controlled by slew rate, and rising edges are governed by the RC time constant formed by the pull-up resistor and the SCL capacitance. See SCL CLKDIV for details.

## Serial Data Signal (SDA)

This is a bidirectional signal on which serial data is transmitted or received depending on the direction of the transfer.

## TWI Internal Interface

The peripheral bus interface supports the transfer of 16-bit wide data and is used by the processor in the support of register and FIFO buffer reads and writes. The TWI internal interface is comprised of the blocks described below.

**Register Block.** Contains all control and status bits and reflects what can be written or read as outlined by the programming model. Status bits can be updated by their respective functional blocks.

**FIFO Buffer.** Configured as a 1-byte-wide 2-deep transmit FIFO buffer and a 1-byte-wide 2-deep receive FIFO buffer.

**Transmit Shift Register.** Serially shifts its data out externally off chip. The output can be controlled for generation of acknowledgments or it can be manually overwritten.

**Receive Shift Register.** Receives its data serially from off chip. The receive shift register is 1 byte wide and data received can either be transferred to the FIFO buffer or used in an address comparison.

**Address Compare Block.** Supports address comparison in the event the TWI controller module is accessed as a slave.

**Prescaler Block.** Must be programmed to generate a 10 MHz time reference relative to the system clock. This time base is used for filtering of data and timing events specified by the electrical data sheet (See the Philips specification), as well as for SCL clock generation.

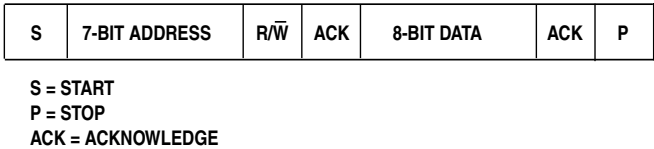
**Clock Generation Module.** Generates an external SCL clock when in master mode. It includes the logic necessary for synchronization in a multi-master clock configuration and clock stretching when configured in slave mode.

## TWI Architectural Concepts

The TWI controller follows the transfer protocol of the Philips I<sup>2</sup>C Bus Specification version 2.1 dated January 2000.

## TWI Protocol

The following figure shows a simple complete transfer.



To better understand the mapping of TWI controller register contents to a basic transfer, The following figure details the same transfer as above noting the corresponding TWI controller bit names. In this illustration, the TWI controller successfully transmits one byte of data. The slave has acknowledged both address and data.



Figure 20-2: Data Transfer with Bit Illustration

Clock Generation and Synchronization

The TWI controller implementation only issues a clock during master mode operation and only at the time a transfer has been initiated. If arbitration for the bus is lost, the serial clock output immediately three-states. If multiple clocks attempt to drive the serial clock line, the TWI controller synchronizes its clock with the other remaining clocks. This is shown in the figure below.

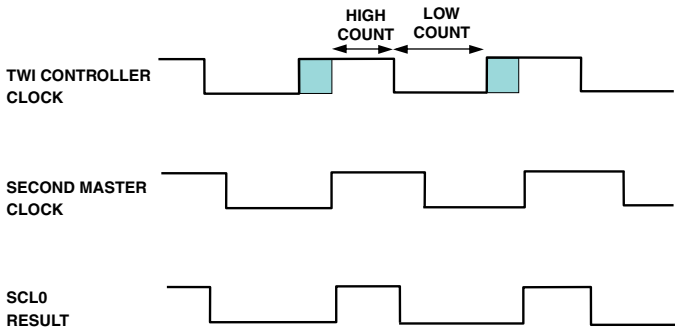


Figure 20-3: Clock Synchronization

The TWI controller serial clock (SCL) output follows these rules:

- Once the clock high (CLKHI) count is complete, the serial clock output is driven low and the clock low (CLKLO) count begins.
- Once the clock low count is complete, the serial clock line is three-stated, allowing the external pull-up resistor to pull SCL high, and the clock synchronization logic enters into a delay mode (shaded area) until the SCL line is detected at logic 1 level. At this time the clock high count begins.



## Bus Arbitration

The TWI controller initiates a master mode transmission only when the bus is idle. If the bus is idle and two masters initiate a transfer, arbitration for the bus begins. This is shown in the figure below.

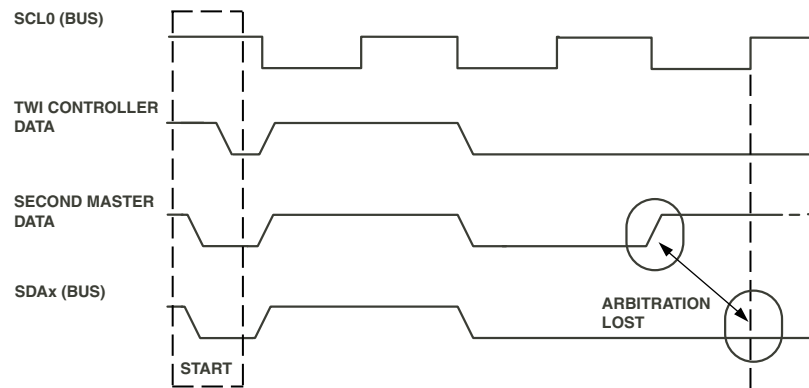


**Figure 20-4:** Bus Arbitration

The TWI controller monitors the serial data bus (SDA) while SCL is high and if SDA is determined to be an active logic 0 level while the TWI controller's data is a logic 1 level, the TWI controller has lost arbitration and ends generation of clock and data. Note that arbitration is not only performed at serial clock edges, but also during the entire time SCL is high.

## Start and Stop Conditions

Start and stop conditions involve serial data transitions while the serial clock is a logic 1 level. The TWI controller generates and recognizes these transitions. Typically start and stop conditions occur at the beginning and at the conclusion of a transmission with the exception repeated start “combined” transfers, as shown in the figure below.

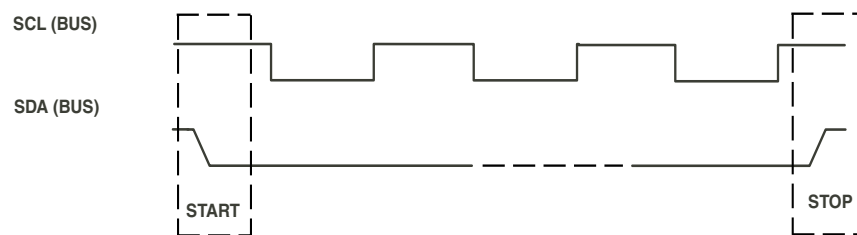


**Figure 20-5:** Start and Stop Conditions

The TWI controller's special case start and stop conditions include the following.

- Controller addressed as a slave-receiver. If the master asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (SCOMP).
- Controller addressed as a slave-transmitter. If the master asserts a stop condition during the data phase of a transfer, the TWI controller concludes the transfer (SCOMP) and indicates a slave transfer error (SERR).

- Controller as a master-transmitter or master-receiver. If the stop bit is set during an active master transfer, the TWI controller issues a stop condition as soon as possible avoiding any error conditions (as if data transfer count had been reached).

## General Call Support

The TWI controller always decodes and acknowledges a general call address if it is enabled as a slave and if general call is enabled. General call addressing (0x00) is indicated by the `GCALL` bit being set and by nature of the transfer the TWI controller is a slave-receiver. If the data associated with the transfer is to be NAK'ed, the NAK bit can be set. If the TWI controller is to issue a general call as a master-transmitter the appropriate address and transfer direction can be set along with loading transmit FIFO data.

## Fast Mode

Fast mode essentially uses the same mechanics as the standard mode of operation. It is the electrical specifications and timing that are most affected. When fast mode is enabled (FAST) timing is modified to meet the electrical requirements as described below.

- Serial data rise times before arbitration evaluation ( $t_r$ )
- Stop condition set-up time from serial clock to serial data ( $t_{SUSTO}$ )
- Bus free time between a stop and start condition ( $t_{BUF}$ )

## TWI Operating Modes

The TWI has two modes of operation, *repeated start* and *clock stretching*. These are described in the following sections.

### Repeated Start

A repeated start condition is the absence of a stop condition between two transfers. The two transfers can be of any direction type. Examples include a transmit followed by a receive, or a receive followed by a transmit. The following sections guide the programmer in developing a service routine.

### Transmit Receive Repeated Start

The following figure shows a repeated start followed by a data receive sequence.

The following tasks are performed at each interrupt.

- TXSERV interrupt. This interrupt was generated due to a FIFO access. Since this is the last byte of this transfer, the `FIFOSTAT` register indicates the transmit FIFO is empty. When read, `DCNT` is zero. Set the `RSTART` bit to indicate a repeated start and set the `DIR` bit if the following transfer will be a data receive.
- MCOMP interrupt. This interrupt was generated because all data has been transferred (`DCNT = 0`). If no errors were generated, a start condition is initiated. Clear the `RSTART` bit and program the `DCNT` with the desired number of bytes to receive.
- RXSERV interrupt. This interrupt is generated due to the arrival of a byte in the receive FIFO. Simple data handling is all that is required.
- MCOMP interrupt. The transfer is complete.

## Receive Transmit Repeated Start

The following figure illustrates a repeated start data receive followed by a data transmit sequence. The tasks performed at each interrupt are:

- RXSERV interrupt. This interrupt is generated due to the arrival of a data byte in the receive FIFO. Set the `RSTART` bit to indicate a repeated start and clear the `MDIR` bit if the following transfer will be a data transmit.
- MCOMP interrupt. This interrupt has occurred due to the completion of the data receive transfer. If no errors were generated, a start condition is initiated. Clear the `RSTART` bit and program the `DCNT` with the desired number of bytes to transmit.
- TXSERV interrupt. This interrupt is generated due to a FIFO access. Simple data handling is all that is required.
- MCOMP interrupt. The transfer is complete.

**NOTE:** There is no timing constraint to meet the above conditions—the user can program the bits as required. Refer to “Clock Stretching During Repeated Start Condition” section for more on how the controller stretches the clock during Repeated Start transfers.

## Clock Stretching

Clock stretching is an added function of the TWI controller in master mode operation. This new behavior uses self-induced stretching of the I<sup>2</sup>C clock while waiting on servicing interrupts. Stretching is done automatically by the hardware and no programming is required for this. The TWI Controller as master supports three modes of clock stretching: clock stretching during FIFO Underflow, clock stretching during FIFO overflow and clock stretching during repeated start condition.

Clock Stretching During FIFO Underflow

During a master mode transmit, an interrupt is generated at the instant the transmit FIFO becomes empty. At this time, the most recent byte begins transmission. If the TXSERV interrupt is not serviced, the concluding acknowledge phase of the transfer is stretched. Stretching of the clock continues until new data bytes are written to the transmit FIFO (TXDATA8 or TXDATA16). No other action is required to release the clock and continue the transmission. This behavior continues until the transmission is complete ( DCNT = 0) at which time the transmission is concluded (MCOMP) as shown in the following figure and table.

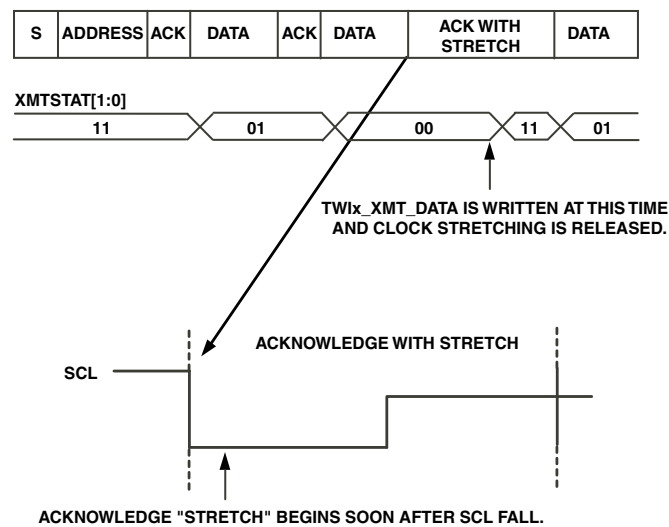


Figure 20-6: Clock Stretching during FIFO Underflow

| TWI Controller                                            | Processor                                                             |
|-----------------------------------------------------------|-----------------------------------------------------------------------|
| Interrupt: XMTSERV – Transmit FIFO buffer is empty.       | Acknowledge: Clear interrupt source bits. Write transmit FIFO buffer. |
| ...                                                       | ...                                                                   |
| Interrupt: MCOMP – Master transmit complete (DCNT= 0x00). | Acknowledge: Clear interrupt source bits.                             |

Clock Stretching During FIFO Overflow

During a master mode receive, an interrupt is generated at the instant the receive FIFO becomes full. It is during the acknowledge phase of this received byte that clock stretching begins. No attempt is made to initiate the reception of an additional byte. Stretching of the clock continues until the data bytes previously received are read from the receive FIFO buffer (RXDATA8 or RXDATA16). No other action is required to release the clock and continue the reception of data. This behavior continues until the reception is

complete (DCNT = 0x00) at which time the reception is concluded (MCOMP) as shown in the following figure and table.

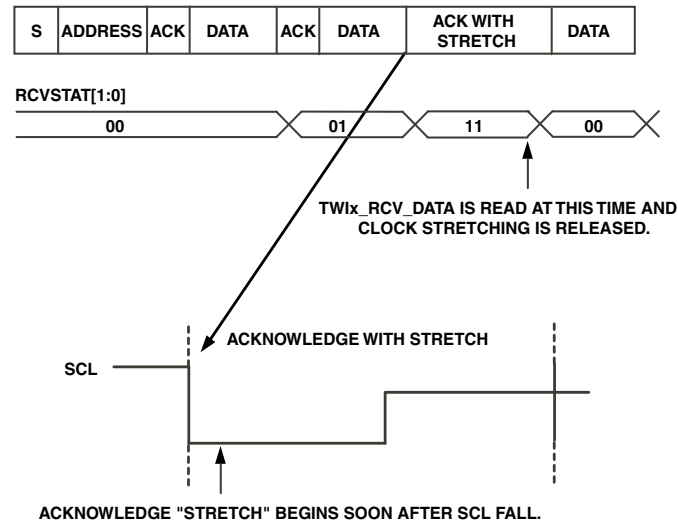


**Figure 20-7:** Clock Stretching During FIFO Overflow

| TWI Controller                                    | Processor                                                           |
|---------------------------------------------------|---------------------------------------------------------------------|
| Interrupt: RCVSERV – Receive FIFO buffer is full. | Acknowledge: Clear interrupt source bits. Read receive FIFO buffer. |
| ...                                               | ...                                                                 |
| Acknowledge: Clear interrupt source bits.         | Interrupt: MCOMP – Master receive complete.                         |

## Clock Stretching During Repeated Start

The repeated start feature in I2C protocol requires transitioning between two subsequent transfers. With the use of clock stretching, the task of managing transitions becomes simpler and becomes common to all transfer types.

Once an initial TWI master transfer has completed (transmit or receive) the clock initiates a stretch during the repeated start phase between transfers. Concurrent with this event the initial transfer generates a transfer complete interrupt (MCOMP) to signify the initial transfer has completed (DCNT = 0). This initial transfer is handled without any special bit setting sequences or timing. The clock stretching logic described above applies here. With no system related timing constraints the subsequent transfer (receive or transmit) is setup and activated. This sequence can be repeated as many times as required to string a series of repeated start transfers together. This is shown in the following figure and table.

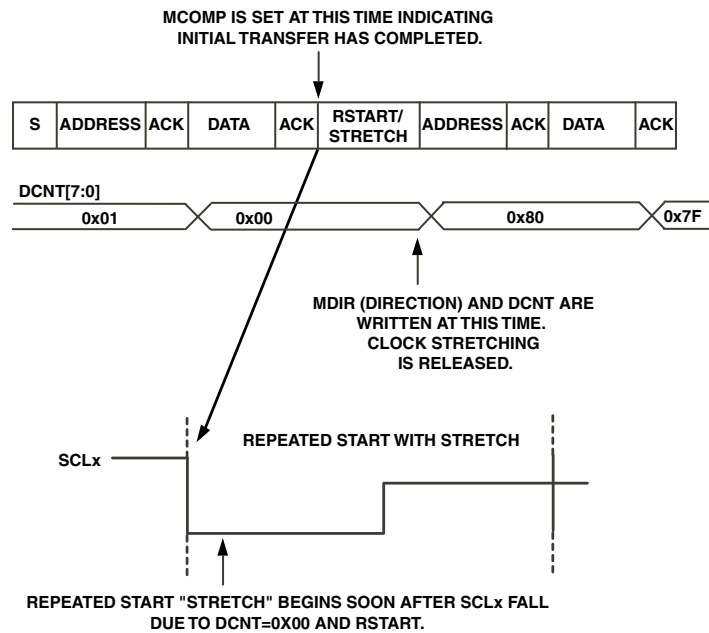


Figure 20-8: Clock Stretching during Repeated Start Condition

| TWI Controller                                                                                                        | Processor                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Interrupt: MCOMP – Initial transmit has completed and DCNT = 0x00. Note: transfer in progress, RSTART previously set. | Acknowledge: Clear interrupt source bits. Write TWIx_MASTER_CTL, setting MDIR (receive), clearing RSTART, and setting new DCNT value (nonzero). |
| Interrupt: RCVSERV – Receive FIFO is full.                                                                            | Acknowledge: Clear interrupt source bits. Read receive FIFO buffer.                                                                             |
| ...                                                                                                                   | ...                                                                                                                                             |
| Interrupt: MCOMP – Master receive complete                                                                            | Acknowledge: Clear interrupt source bits.                                                                                                       |

## TWI Programming Model

The topics in this section provide information on the basic programming steps required to set up and run the two wire interface.

The following sections provide general setup, and master and slave mode programming steps.

## TWI General Setup

General setup refers to register writes that are required for both slave mode and master mode operation.

General setup should be performed before either the master or slave enable bits are set.

1. Program the CTL register to enable the TWI controller and set the prescale value.
2. Program the prescale value to the binary representation of  $f_{\text{SCLK}}/10$  MHz. All values should be rounded up to the next whole number.
3. Set the EN bit in the CTL register.

*RESULT:*

Once the TWI controller is enabled a bus busy condition may be detected. This condition should clear after  $t_{\text{BUF}}$  has expired assuming no additional bus activity has been detected.

## Slave Mode

When enabled, slave mode operation supports both receive and transmit data transfers.

It is not possible to enable only one data transfer direction and not acknowledge (NAK) the other. This is reflected in the following setup.

1. Program the SLVADDR register. The appropriate 7 bits are used in determining a match during the address phase of the transfer.
2. Program the TXDATA8 or TXDATA16 registers. These are the initial data values to be transmitted in the event the slave is addressed and a transmit is required. This is an optional step. If no data is written and the slave is addressed and a transmit is required, the serial clock (SCL) is stretched and an interrupt is generated until data is written to the transmit FIFO.
3. Program the IMASK register. Enable bits are associated with the desired interrupt sources. As an example, programming the value 0x000F results in an interrupt output to the processor in the event that a valid address match is detected, a valid slave transfer completes, a slave transfer has an error, a subsequent transfer has begun yet the previous transfer has not been serviced.
4. Program the SLVCTL register. Ultimately this prepares and enables slave mode operation. As an example, programming the value 0x0005 enables slave mode operation, requires 7-bit addressing, and indicates that data in the transmit FIFO buffer is intended for slave mode transmission.

*RESULT:*

The following table and flow diagram shows what the interaction between the TWI controller and the processor might look like using this example.

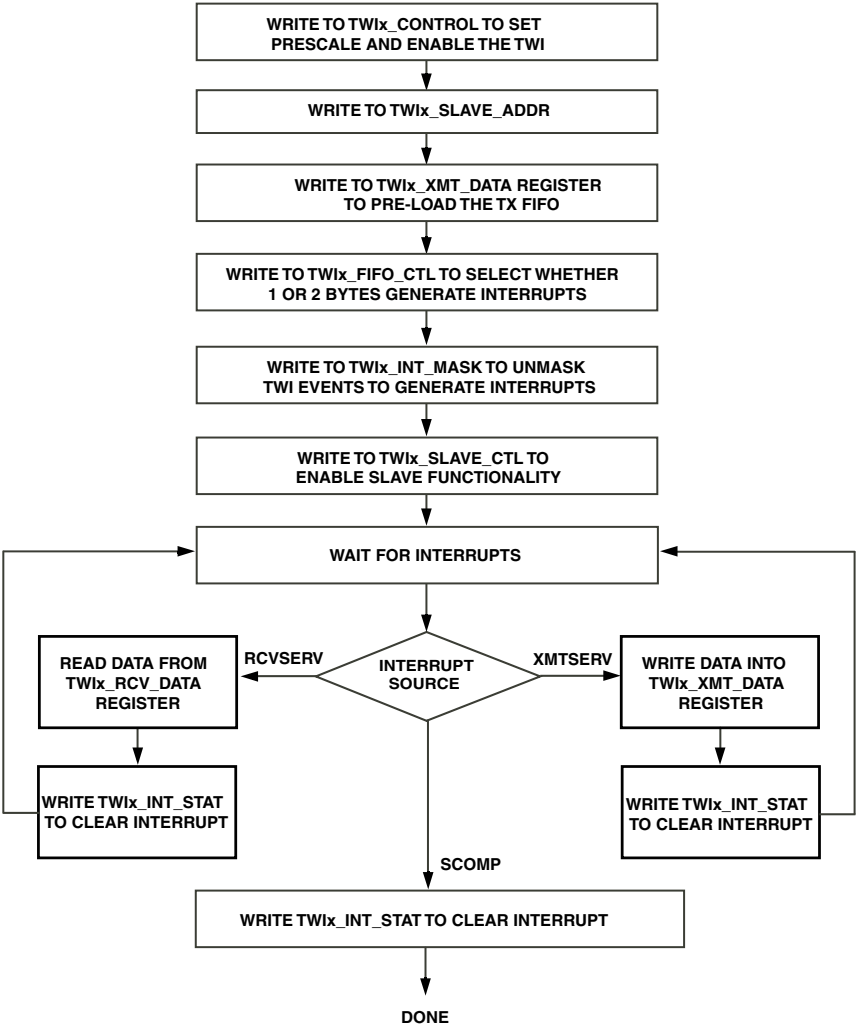


Figure 20-9: TWI Slave Mode Program Flow

Table 20-3: Slave Mode Interaction

| TWI Controller                                 | Processor                                                                                |
|------------------------------------------------|------------------------------------------------------------------------------------------|
| Interrupt: SINIT – Slave transfer in progress. | Acknowledge: Clear interrupt source bits.                                                |
| Interrupt: RCVSERV – Receive buffer is full.   | Acknowledge: Clear interrupt source bits. Read TWIx_FIFO_STAT. Read receive FIFO buffer. |
| ...                                            | ...                                                                                      |
| Interrupt: SCOMP – Slave transfer complete.    | Acknowledge: Clear interrupt source bits. Read receive FIFO buffer.                      |



## Master Mode Clock Setup

Master mode operation is set up and executed on a per-transfer basis.

An example of programming steps for a receive and for a transmit are given separately in following sections. The clock setup programming step listed here is common to both transfer types.

1. Program `CLKDIV` to define the minimum `CLKHI` duration and minimum `CLKLO` duration.

*RESULT:*

The `CLKHI` and `CLKLO` fields of the `CLKDIV` register are not intended to guarantee a certain frequency. Rather, they guarantee a certain minimum high and low duration for SCL. Falling edges are controlled by slew rate, and rising edges are governed by the RC time constant formed by the pull-up resistor and the SCL capacitance. See SCL clock divider register (`CLKDIV`) for details.

## Master Mode Transmit

Follow these programming steps for a single master mode transmit:

1. Program the `MSTRADDR` register. This defines the address transmitted during the address phase of the transfer.
2. Program the `TXDATA8` or `TXDATA16` register. This is the initial data transmitted. It is considered an error to complete the address phase of the transfer and not have data available in the transmit FIFO buffer.
3. Program the `FIFOCTL` register. Indicate if the transmit FIFO buffer interrupts should occur with each byte transmitted (8-bits) or with each two bytes transmitted (16-bits).
4. Program the `IMSK` register. Enable bits associated with the desired interrupt sources. As an example, programming the value `0x0030` results in an interrupt output to the processor in the event that the master transfer completes, and the master transfer has an error.
5. Program the `MSTRCTL` register. This prepares and enables master mode operation. As an example, programming the value `0x0201` enables master mode operation, generates a 7-bit address, sets the direction to master-transmit, uses standard mode timing, and transmits 8 data bytes before generating a Stop condition.

*RESULT:*

**Table 20-4:** Master Mode Transmit Setup Interaction

| TWI Controller                                              | Processor                                                             |
|-------------------------------------------------------------|-----------------------------------------------------------------------|
| Interrupt: <code>XMTSERV</code> – Transmit buffer is empty. | Acknowledge: Clear interrupt source bits. Write transmit FIFO buffer. |

**Table 20-4:** Master Mode Transmit Setup Interaction (Continued)

| TWI Controller                               | Processor                                 |
|----------------------------------------------|-------------------------------------------|
| ...                                          | ...                                       |
| Interrupt: MCOMP – Master transfer complete. | Acknowledge: Clear interrupt source bits. |

## Master Mode Receive

Follow these programming steps for a single master mode receive.

1. Program the MSTRADDR register. This defines the address transmitted during the address phase of the transfer.
2. Program the FIFOCTL register. Indicate if receive FIFO buffer interrupts should occur with each byte received (8-bits) or with each two bytes received (16-bits).
3. Program the IMSK register. Enable bits associated with the desired interrupt sources. For example, programming the value 0x0030 results in an interrupt output to the processor in the event that the master transfer completes, and the master transfer has an error.
4. Program the MSTRCTL register. This prepares and enables master mode operation. As an example, programming the value 0x0205 enables master mode operation, generates a 7-bit address, sets the direction to master-receive, uses standard mode timing, and receives 8 data bytes before generating a Stop condition.

### RESULT:

The following table shows what the interaction between the TWI controller and the processor might look like using this example.

**Table 20-5:** Master Mode Receive Setup Interaction

| TWI Controller                               | Processor                                                           |
|----------------------------------------------|---------------------------------------------------------------------|
| Interrupt: RCVSERV – Receive buffer is full. | Acknowledge: Clear interrupt source bits. Read receive FIFO buffer. |
| ...                                          | ...                                                                 |
| Interrupt: MCOMP – Master transfer complete. | Acknowledge: Clear interrupt source bits. Read receive FIFO buffer. |

## Master Mode Program Flow

The following figure shows the program for the TWI in master mode.

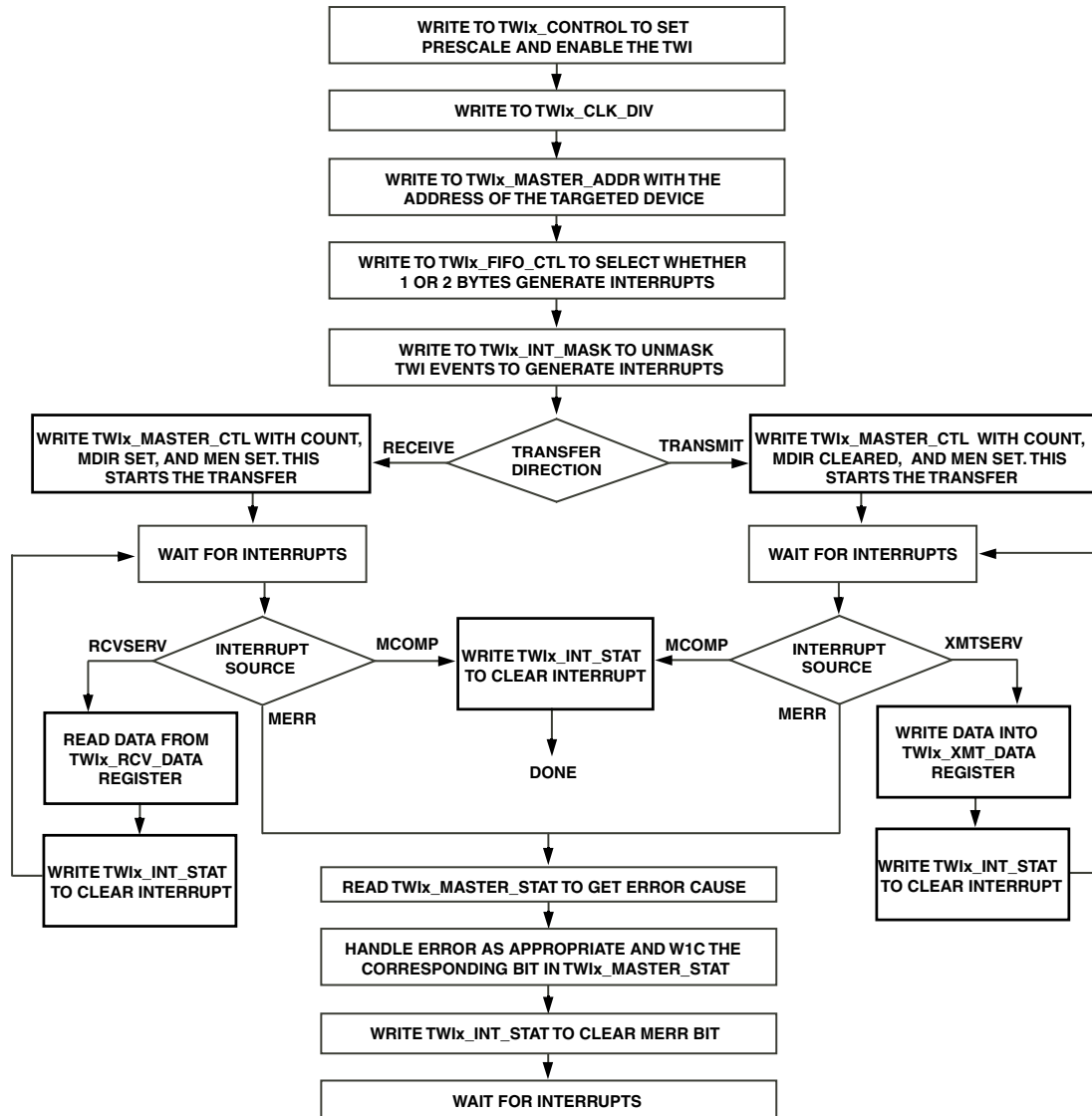


Figure 20-10: Master Mode Program Flow

## TWI Register Descriptions

2-Wire Interface (TWI) contains the following registers.

**Table 20-6:** TWI Module Registers

| Name         | Description                   |
|--------------|-------------------------------|
| TWI_CLKDIV   | SCL Clock Divider Register    |
| TWI_CTL      | Control Register              |
| TWI_SLVCTL   | Slave Mode Control Register   |
| TWI_SLVSTAT  | Slave Mode Status Register    |
| TWI_SLVADDR  | Slave Mode Address Register   |
| TWI_MSTRCTL  | Master Mode Control Registers |
| TWI_MSTRSTAT | Master Mode Status Register   |
| TWI_MSTRADDR | Master Mode Address Register  |
| TWI_ISTAT    | Interrupt Status Register     |
| TWI_IMSK     | Interrupt Mask Register       |
| TWI_FIFOCTL  | FIFO Control Register         |
| TWI_FIFOSTAT | FIFO Status Register          |
| TWI_TXDATA8  | Tx Data Single-Byte Register  |
| TWI_TXDATA16 | Tx Data Double-Byte Register  |
| TWI_RXDATA8  | Rx Data Single-Byte Register  |
| TWI_RXDATA16 | Rx Data Double-Byte Register  |

## SCL Clock Divider Register

During master mode operation, the TWI\_CLKDIV holds values, which the TWI uses to create the high and low durations of the serial clock (SCL). The clock signal SCL is an output in master mode and an input in slave mode. The values in the TWI\_CLKDIV.CLKLO and TWI\_CLKDIV.CLKHI fields add up to the CLKDIV value the following equation.

$$\text{CLKDIV} = \text{TWI SCL period} / 10 \text{ MHz time reference}$$

Serial clock frequencies can vary from 400 KHz to less than 20 KHz. The resolution of the clock generated is 1/10 MHz or 100 ns. For example, for an SCL of 400 KHz (period = 1/400 KHz = 2500 ns) and an internal time reference of 10 MHz (period = 100 ns):

$$\text{CLKDIV} = 2500 \text{ ns} / 100 \text{ ns} = 25$$

For an SCL with a 30% duty cycle, use `TWI_CLKDIV.CLKLO = 17` and `TWI_CLKDIV.CLKHI = 8`.

**TWI\_CLKDIV: SCL Clock Divider Register - R/W**

Reset = 0x0000

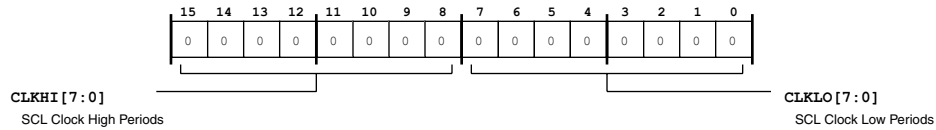


**Figure 20-11:** TWI\_CLKDIV Register Diagram

**Table 20-7:** TWI\_CLKDIV Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | CLKHI    | SCL Clock High Periods.<br>The <code>TWI_CLKDIV.CLKHI</code> specifies the number of 10 MHz time reference periods the serial clock (SCL) waits before a new clock low period begins, assuming a single master. |
| 7:0     | CLKLO    | SCL Clock Low Periods.<br>The <code>TWI_CLKDIV.CLKLO</code> specifies the number of internal time reference periods the serial clock (SCL) is held low.                                                         |

## Control Register

The `TWI_CTL` enables the TWI, establishes a relationship between the system clock (SCLK) and the TWI controller's internally timed events, and enables SCCB compatibility.

**TWI\_CTL: Control Register - R/W**

Reset = 0x0000

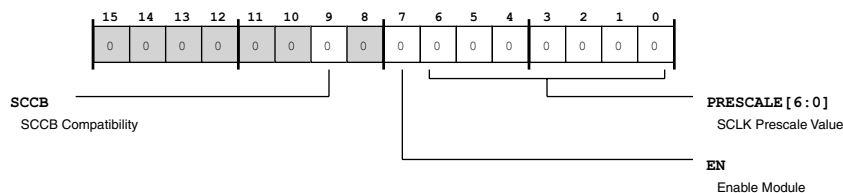


**Figure 20-12:** TWI\_CTL Register Diagram

**Table 20-8:** TWI\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                      |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | SCCB     | SCCB Compatibility.<br>The <code>TWI_CTL.SCCB</code> enables SCCB compatible operation for the TWI. SCCB compatibility is an optional feature and should not be used in an I <sup>2</sup> C bus system. When this feature is enabled, all slave asserted acknowledgement bits are ignored by this master. This feature is valid only during transfers where the TWI is mastering an SCCB bus. Slave mode transfers should be avoided when this feature is enabled because the TWI controller always generates an acknowledge in slave mode.     |                                                                                                                                                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Disable SCCB compatibility<br>When disabled, Master transfers are not SCCB compatible.                                                               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Enable SCCB compatibility<br>When enabled, Master transfers are SCCB compatible. All slave-asserted acknowledgement bits are ignored by this master. |
| 7       | EN       | Enable Module.<br>The <code>TWI_CTL.EN</code> enables TWI controller operation for either master and/or slave mode of operation. It is recommended that this bit be set at the time <code>TWI_CTL.PRESCALE</code> is initialized and remain set. This method guarantees accurate operation of bus busy detection logic.                                                                                                                                                                                                                         |                                                                                                                                                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Disable                                                                                                                                              |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Enable                                                                                                                                               |
| 6:0     | PRESCALE | SCLK Prescale Value.<br>The <code>TWI_CTL.PRESCALE</code> holds the pre-scaled value for the TWI internal time reference. This reference is derived from SCLK according to the formula:<br>$TWI\_CTL.PRESCALE = f_{SCLK}/10MHz$<br>The <code>TWI_CTL.PRESCALE</code> specifies the number of system clock (SCLK) periods used in the generation of one internal time reference. The value of <code>TWI_CTL.PRESCALE</code> must be set to create an internal time reference with a period of 10 MHz. It is represented as a 7-bit binary value. |                                                                                                                                                      |

## Slave Mode Control Register

The `TWI_SLVCTL` controls the logic associated with slave mode operation. Settings in this register do not affect master mode operation and should not be modified to control master mode functionality.

**TWI\_SLVCTL: Slave Mode Control Register - R/W**

Reset = 0x0000

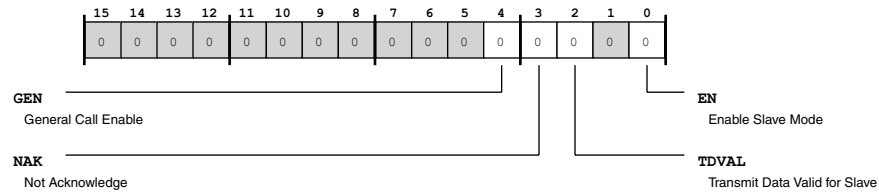


**Figure 20-13:** TWI\_SLVCTL Register Diagram

**Table 20-9:** TWI\_SLVCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | GEN      | General Call Enable.<br>The <code>TWI_SLVCTL.GEN</code> enables general call address matching. When enabled, a general call slave receive transfer is accepted. All status and interrupt source bits associated with transfers are updated. Note that general call address detection is available only when slave mode is enabled. |
|         |          | 0      Disable General Call Matching                                                                                                                                                                                                                                                                                               |
|         |          | 1      Enable General Call Matching                                                                                                                                                                                                                                                                                                |
| 3       | NAK      | Not Acknowledge.<br>The <code>TWI_SLVCTL.NAK</code> directs the TWI to generate a NAK (if set) or an ACK (if cleared) at the conclusion of data transfer for slave receive. For NAK, the slave is still considered to be addressed at the conclusion of transfer.                                                                  |
|         |          | 0      Generate ACK                                                                                                                                                                                                                                                                                                                |
|         |          | 1      Generate NAK                                                                                                                                                                                                                                                                                                                |

Table 20-9: TWI\_SLVCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | TDVAL    | Transmit Data Valid for Slave.<br>The TWI_SLVCTL.TDVAL selects whether the data in the transmit FIFO is available (valid) for slave transmission (TWI_SLVCTL.TDVAL set). If the FIFO data is not available (invalid) for slave transmission (TWI_SLVCTL.TDVAL cleared), the data in the transmit FIFO is for master mode transmits, and the data is not allowed to be used during a slave transmit; the transmit FIFO is treated as if it is empty. |
|         |          | 0 Data Invalid for Slave Tx                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Data Valid for Slave Tx                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 0       | EN       | Enable Slave Mode.<br>The TWI_SLVCTL.EN enables slave operation. Enabling slave and master modes of operation concurrently is allowed. If disabled, no attempt is made to identify a valid address. If TWI_SLVCTL.EN is cleared during a valid transfer, clock stretching ceases, the serial data line is released, and the current byte is not acknowledged.                                                                                       |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                            |

Slave Mode Status Register

During and at the conclusion of register slave mode transfers, the TWI\_SLVSTAT holds information on the current transfer. Generally slave mode status bits are not associated with the generation of interrupts. Master mode operation does not affect slave mode status bits.

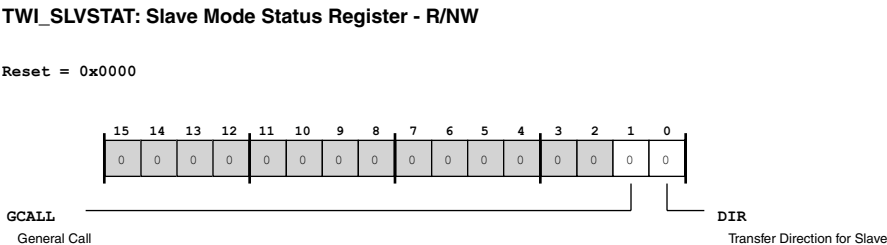


Figure 20-14: TWI\_SLVSTAT Register Diagram



**Table 20-10:** TWI\_SLVSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | GCALL    | General Call.<br>The <code>TWI_SLVSTAT.GCALL</code> indicates whether or not--at the time of addressing--the address was determined to be a general call. This bit self clears if slave mode is disabled ( <code>TWI_SLVCTL.EN = 0</code> ).                              |
|         |          | 0                      Not a General Call Address                                                                                                                                                                                                                         |
|         |          | 1                      General Call Address                                                                                                                                                                                                                               |
| 0       | DIR      | Transfer Direction for Slave.<br>The <code>TWI_SLVSTAT.DIR</code> indicates whether--at the time of addressing--the transfer direction was determined to be slave transmit or receive. This bit self clears if slave mode is disabled ( <code>TWI_SLVCTL.EN = 0</code> ). |
|         |          | 0                      Slave Receive                                                                                                                                                                                                                                      |
|         |          | 1                      Slave Transmit                                                                                                                                                                                                                                     |

## Slave Mode Address Register

The `TWI_SLVADDR` holds the slave mode address, which is the valid address to which the slave-enabled TWI controller responds. The TWI controller compares this value with the received address during the addressing phase of a transfer.

**TWI\_SLVADDR: Slave Mode Address Register - R/W**

Reset = 0x0000

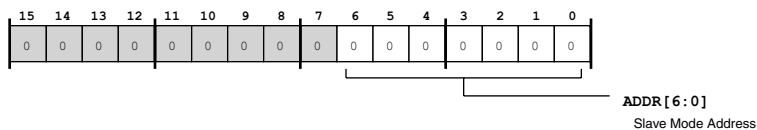


**Figure 20-15:** TWI\_SLVADDR Register Diagram

**Table 20-11:** TWI\_SLVADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 6:0     | ADDR     | Slave Mode Address.     |

Master Mode Control Registers

The TWI\_MSTRCTL controls the logic associated with master mode operation. Bits in this register do not affect slave mode operation and should not be modified to control slave mode functionality.

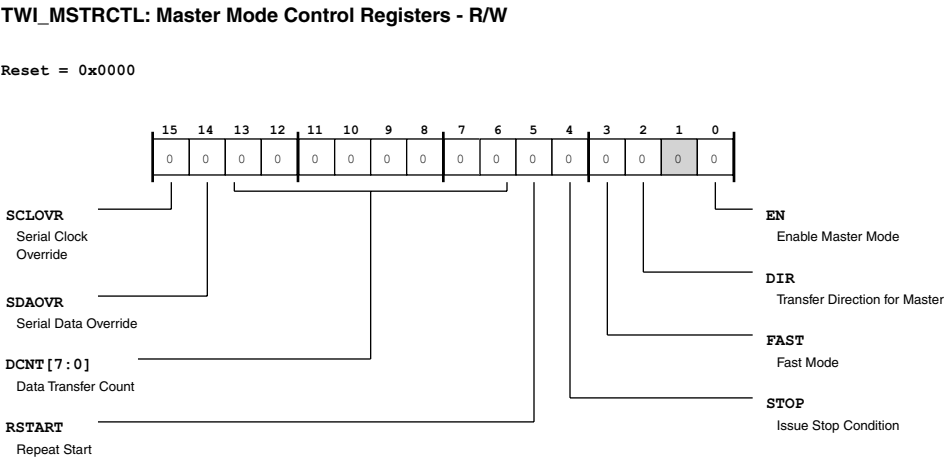


Figure 20-16: TWI\_MSTRCTL Register Diagram

Table 20-12: TWI\_MSTRCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | SCLOVR   | Serial Clock Override.<br>The TWI_MSTRCTL . SCLOVR provides direct control of the serial clock line when required. Normal master and slave mode operation should not require override operation. When TWI_MSTRCTL . SCLOVR is set, the TWI overrides normal serial clock output, driving it to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_MSTRCTL . SCLOVR is cleared, the TWI permits normal serial clock operation under the control of master mode clock generation and slave mode clock stretching logic. |
|         |          | 0 Permit Normal SCL Operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 Override Normal SCL Operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 20-12:** TWI\_MSTRCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | SDAOVR   | Serial Data Override.<br>The TWI_MSTRCTL.SDAOVR provides direct control of the serial data line when required. Normal master and slave mode operation should not require override operation. When TWI_MSTRCTL.SDAOVR is set, the TWI overrides normal serial data operation under the control of the transmit shift register and acknowledge logic, driving serial data output to an active 0 level and overriding all other logic. This state is held until this bit is cleared. When TWI_MSTRCTL.SDAOVR is cleared, the TWI permits normal serial data operation.                                                     |
|         |          | 0 Permit Normal SDA Operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Override Normal SDA Operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 13:6    | DCNT     | Data Transfer Count.<br>The TWI_MSTRCTL.DCNT indicates the number of data bytes to transfer. As each data word is transferred, the TWI decrements this counter. When TWI_MSTRCTL.DCNT decrements to 0, a stop condition is generated. Setting TWI_MSTRCTL.DCNT to 0xFF disables the counter. In this transfer mode, data continues to be transferred until it is concluded by setting the TWI_MSTRCTL.STOP bit. In the event a master transmit is aborted due to a slave data NAK, the value of TWI_MSTRCTL.DCNT equals the number of bytes not sent. The byte which was NAK'ed by the slave is counted as a sent byte. |
| 5       | RSTART   | Repeat Start.<br>The TWI_MSTRCTL.RSTART enables the TWI to issue a repeat start condition at the conclusion of the current transfer (TWI_MSTRCTL.DCNT = 0) and begin the next transfer. The current transfer concludes with updates to the appropriate status and interrupt bits. If errors occurred during the previous transfer, a repeat start does not occur. In the absence of any errors, master enable (TWI_MSTRCTL.EN) does not self clear on a repeat start.                                                                                                                                                   |
|         |          | 0 Disable Repeat Start                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable Repeat Start                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Table 20-12:** TWI\_MSTRCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | STOP     | Issue Stop Condition.<br>The <code>TWI_MSTRCTL.STOP</code> directs the TWI to issue a stop condition. The transfer concludes as soon as possible avoiding any error conditions (as if data transfer count had been reached). At that time, the <code>TWI_IMSK</code> is updated along with any associated status bits.                                                                                                                                                                            |
|         |          | 0 Permit Normal Operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Issue Stop                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 3       | FAST     | Fast Mode.<br>The <code>TWI_MSTRCTL.FAST</code> selects whether the TWI operates in fast mode or standard mode. In fast mode, the TWI uses timing specifications for transfers at up to 400K bits/s. In standard mode, the TWI uses timing specifications for transfers at up to 100K bits/s.                                                                                                                                                                                                     |
|         |          | 0 Select Standard Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Select Fast Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 2       | DIR      | Transfer Direction for Master.<br>The <code>TWI_MSTRCTL.DIR</code> selects the transfer direction for the TWI as master initiated receive or transmit.                                                                                                                                                                                                                                                                                                                                            |
|         |          | 0 Master Transmit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Master Receive                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 0       | EN       | Enable Master Mode.<br>The <code>TWI_MSTRCTL.EN</code> enables master mode functionality. A start condition is generated if the bus is idle. This bit self clears at the completion of a transfer, including transfers terminated due to errors. If disabled (bit cleared) during operation, the transfer is aborted, and all logic associated with master mode transfers are reset. Serial data and serial clock (SDA, SCL) are no longer driven. Write-1-to-clear status bits are not affected. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## Master Mode Status Register

The `TWI_MSTRSTAT` holds information during master mode transfers and at their conclusion. Generally, master mode status bits are not directly associated with the generation of interrupts, but these bits offer information on the current transfer. Slave mode operation does not affect master mode status bits.

Note that while `TWI_MSTRSTAT.SCLSEN` is set (this condition could be due to having no pull-up resistor on `TWI_SCL` or another agent is driving `TWI_SCL` low), the acknowledge bits (`TWI_MSTRSTAT.ANAK` and `TWI_MSTRSTAT.DNAK`) do not update. This result occurs because the acknowledge conditions are sampled during the high phase of `TWI_SCL`.

**TWI\_MSTRSTAT: Master Mode Status Register - R/NW**

Reset = 0x0000

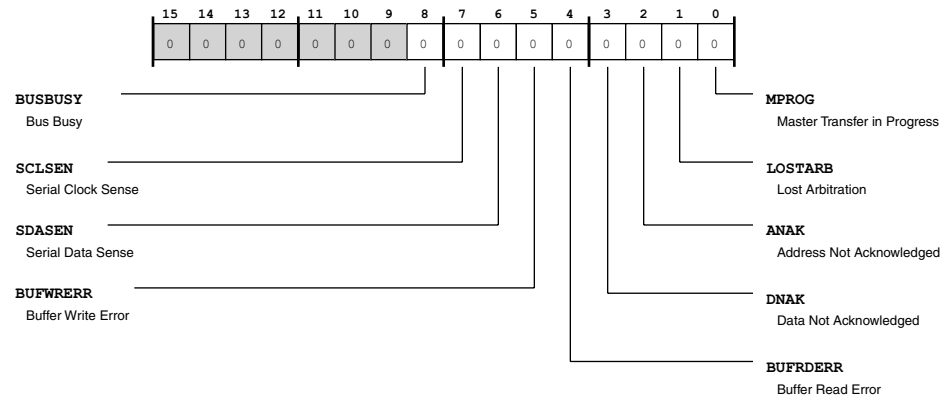


**Figure 20-17:** TWI\_MSTRSTAT Register Diagram

**Table 20-13:** TWI\_MSTRSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                          |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | BUSBUSY  | Bus Busy.<br>The <code>TWI_MSTRSTAT.BUSBUSY</code> indicates whether the bus is currently busy or free. This indication is not limited to only this device but is for all devices. On a start condition, the setting of the register value is delayed due to the input filtering. On a stop condition the clearing of the register value occurs after $t_{BUF}$ . |                                                                                                                                                          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                 | Bus Free<br>The bus is free. The clock and data bus signals have been inactive for the appropriate bus free time.                                        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                 | Bus Busy<br>The bus is busy. Clock or data activity has been detected.                                                                                   |
| 7       | SCLSEN   | Serial Clock Sense.<br>The <code>TWI_MSTRSTAT.SCLSEN</code> indicates the active or inactive state of the serial clock. Use this status bit when direct sensing of the serial clock line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal master and slave mode operation should not require this feature.            |                                                                                                                                                          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                 | SCL Inactive "One"<br>An inactive "one" is being sensed on the serial clock.                                                                             |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                 | SCL Active "Zero"<br>An active "zero" is being sensed on the serial clock. The source of the active driver is not known and can be internal or external. |

**Table 20-13:** TWI\_MSTRSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | SDASEN   | Serial Data Sense.<br>The TWI_MSTRSTAT.SDASEN indicates the active or inactive status of the serial data. Use this status bit when direct sensing of the serial data line is required. The register value is delayed due to the input filter (nominally 50 ns). Normal master and slave mode operation should not require this feature. |
|         |          | 0<br>SDA Inactive "One"<br>An inactive "one" is currently being sensed on the serial data line.                                                                                                                                                                                                                                         |
|         |          | 1<br>SDA Active "Zero"<br>An active "zero" is currently being sensed on the serial data line. The source of the active driver is not known and can be internal or external.                                                                                                                                                             |
| 5       | BUFWRERR | Buffer Write Error.<br>The TWI_MSTRSTAT.BUFWRERR indicates whether the current master transfer was aborted due to a receive buffer write error. The receive buffer and receive shift register were both full at the same time. This bit is W1C.                                                                                         |
|         |          | 0<br>No Status                                                                                                                                                                                                                                                                                                                          |
|         |          | 1<br>Buffer Write Error                                                                                                                                                                                                                                                                                                                 |
| 4       | BUFRDERR | Buffer Read Error.<br>The TWI_MSTRSTAT.BUFRDERR indicates whether the current master transfer was aborted due to the detection of a NAK during data transmission. This bit is W1C.                                                                                                                                                      |
|         |          | 0<br>No Status                                                                                                                                                                                                                                                                                                                          |
|         |          | 1<br>Buffer Read Error                                                                                                                                                                                                                                                                                                                  |

**Table 20-13:** TWI\_MSTRSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | DNAK     | Data Not Acknowledged.<br>The <code>TWI_MSTRSTAT.DNAK</code> indicates whether the current master transfer was aborted due to the detection of a NAK during data transmission. This bit is W1C.                                                                                                                            |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                           |
|         |          | 1                      Data NAK                                                                                                                                                                                                                                                                                            |
| 2       | ANAK     | Address Not Acknowledged.<br>The <code>TWI_MSTRSTAT.ANAK</code> indicates whether the current master transfer was aborted due to the detection of a NAK during the address phase of the transfer. This bit is W1C.                                                                                                         |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                           |
|         |          | 1                      Address NAK                                                                                                                                                                                                                                                                                         |
| 1       | LOSTARB  | Lost Arbitration.<br>The <code>TWI_MSTRSTAT.LOSTARB</code> indicates whether the current transfer was aborted due to the loss of arbitration with another master. This bit is W1C.                                                                                                                                         |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                           |
|         |          | 1                      Lost Arbitration                                                                                                                                                                                                                                                                                    |
| 0       | MPROG    | Master Transfer in Progress.<br>The <code>TWI_MSTRSTAT.MPROG</code> indicates whether or not a master transfer is in progress. If clear ( <code>TWI_MSTRSTAT.MPROG = 0</code> ), currently no transfer is taking place. This can occur after a transfer is complete or while an enabled master is waiting for an idle bus. |
|         |          | 0                      No Status                                                                                                                                                                                                                                                                                           |
|         |          | 1                      Master Transfer in Progress                                                                                                                                                                                                                                                                         |

## Master Mode Address Register

During the addressing phase of a transfer, the TWI controller, with its master enabled, transmits the contents of `TWI_MSTRADDR`. When programming this register, omit the read/write bit. That is, only the upper 7 bits that make up the slave address should be written to this register. For example, if the slave



address is b#1010000X, where X is the read/write bit, the TWI\_MSTRADDR is programmed with b#1010000, which corresponds to 0x50. When sending out the address on the bus, the TWI controller appends the read/write bit as appropriate based on the state of the TWI\_MSTRCTL.DIR bit.

**TWI\_MSTRADDR: Master Mode Address Register - R/W**

Reset = 0x0000

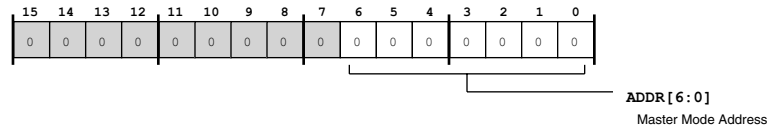


**Figure 20-18:** TWI\_MSTRADDR Register Diagram

**Table 20-14:** TWI\_MSTRADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 6:0     | ADDR     | Master Mode Address.    |

## Interrupt Status Register

The TWI\_ISTAT contains information about functional areas requiring servicing. Many of the bits serve as an indicator to further read and service various status registers. After servicing the interrupt source associated with a bit, the user must clear that interrupt source bit by writing a 1 to it.

**TWI\_ISTAT: Interrupt Status Register - R/WA**

Reset = 0x0000

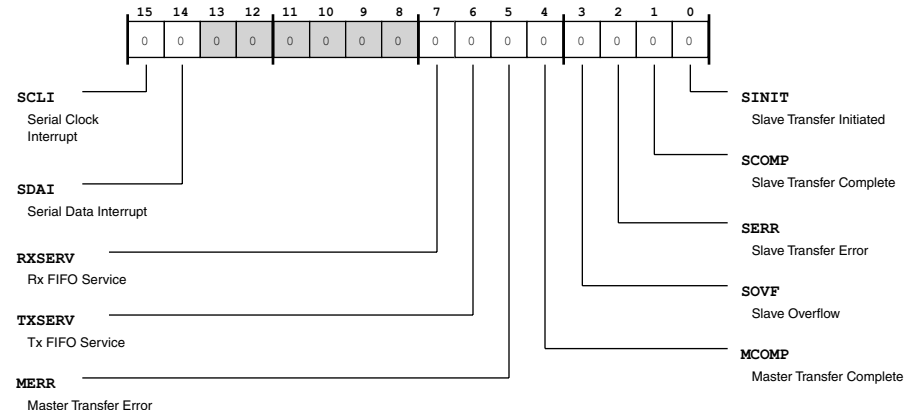


**Figure 20-19:** TWI\_ISTAT Register Diagram

**Table 20-15:** TWI\_ISTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | SCLI     | Serial Clock Interrupt.<br>If the TWI is enabled (TWI_CTL.EN), SCLI is set on a high-to-low transition of the serial clock pin (SCLx). Normally, this bit is not required for I <sup>2</sup> C bus transfers. It will be initially set on an I <sup>2</sup> C transfer and does not require clearing. |
|         |          | 0<br>No Interrupt<br>No transition was detected on the SCLx pin.                                                                                                                                                                                                                                      |
|         |          | 1<br>Interrupt Detected<br>A high-to-low transition was detected on the SCLx pin. This bit is W1C.                                                                                                                                                                                                    |
| 14      | SDAI     | Serial Data Interrupt.<br>If the TWI is enabled (TWI_CTL.EN), SDAI is set on a high-to-low transition of the serial data pin (SDAx). Normally, this bit is not required for I <sup>2</sup> C bus transfers. It will be initially set on an I <sup>2</sup> C transfer and does not require clearing.   |
|         |          | 0<br>No Interrupt<br>No transition was detected on the SDAx pin.                                                                                                                                                                                                                                      |
|         |          | 1<br>Interrupt Detected<br>A high-to-low transition was detected on the SDAx pin. This bit is W1C.                                                                                                                                                                                                    |
| 7       | RXSERV   | Rx FIFO Service.<br>If TWI_FIFOCTL.RXILEN =0, the TWI_ISTAT.RXSERV is set each time the TWI_FIFOSTAT.RXSTAT field is updated to either 01 or 11. If TWI_FIFOCTL.RXILEN =1, the TWI_ISTAT.RXSERV is set each time TWI_FIFOSTAT.RXSTAT is updated to 11.                                                |
|         |          | 0<br>No Interrupt<br>The FIFO does not require servicing, or the TWI_FIFOSTAT.RXSTAT field has not changed since this bit was last cleared.                                                                                                                                                           |
|         |          | 1<br>Interrupt Detected<br>The receive FIFO buffer has one or two 8-bit words of data available to be read.                                                                                                                                                                                           |

**Table 20-15:** TWI\_ISTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                 |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | TXSERV   | Tx FIFO Service.<br>If <code>TWI_FIFOCTL.TXILEN = 0</code> , the <code>TWI_ISTAT.TXSERV</code> is set each time the <code>TWI_FIFOSTAT.TXSTAT</code> field is updated to either 01 or 00. If <code>TWI_FIFOCTL.TXILEN = 1</code> , the <code>TWI_ISTAT.TXSERV</code> is set each time <code>TWI_FIFOSTAT.TXSTAT</code> is updated to 00. |                                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                        | No Interrupt<br>FIFO does not require servicing, or the <code>TWI_FIFOSTAT.TXSTAT</code> field has not changed since this bit was last cleared. |
|         |          | 1                                                                                                                                                                                                                                                                                                                                        | Interrupt Detected<br>The transmit FIFO buffer has one or two 8-bit locations available to be written.                                          |
| 5       | MERR     | Master Transfer Error.<br>The <code>TWI_ISTAT.MERR</code> indicates that a master error has occurred. The conditions surrounding the error are indicated by the master status register ( <code>TWI_MSTRSTAT</code> ).                                                                                                                    |                                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                        | No Interrupt                                                                                                                                    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                        | Interrupt Detected                                                                                                                              |
| 4       | MCOMP    | Master Transfer Complete.<br>The <code>TWI_ISTAT.MCOMP</code> indicates that the initiated master transfer has completed. In the absence of a repeat start, the bus has been released.                                                                                                                                                   |                                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                        | No Interrupt                                                                                                                                    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                        | Interrupt Detected                                                                                                                              |
| 3       | SOVF     | Slave Overflow.<br>The <code>TWI_ISTAT.SOVF</code> indicates that the <code>TWI_ISTAT.SCOMP</code> bit was set at the time a subsequent transfer has acknowledged an address phase. The transfer continues, however, it may be difficult to delineate data of one transfer from another.                                                 |                                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                        | No Interrupt                                                                                                                                    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                        | Interrupt Detected                                                                                                                              |

**Table 20-15:** TWI\_ISTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | SERR     | Slave Transfer Error.<br>The <code>TWI_ISTAT.SERR</code> indicates that a slave error has occurred. A restart or stop condition has occurred during the data receive phase of a transfer. |
|         |          | 0 No Interrupt                                                                                                                                                                            |
|         |          | 1 Interrupt Detected                                                                                                                                                                      |
| 1       | SCOMP    | Slave Transfer Complete.<br>The <code>TWI_ISTAT.SCOMP</code> indicates that the transfer is complete and either a stop, or a restart was detected.                                        |
|         |          | 0 No Interrupt                                                                                                                                                                            |
|         |          | 1 Interrupt Detected                                                                                                                                                                      |
| 0       | SINIT    | Slave Transfer Initiated.<br>The <code>TWI_ISTAT.SINIT</code> indicates whether or not a slave transfer is in progress.                                                                   |
|         |          | 0 No Interrupt<br>A transfer is not in progress, or an address match has not occurred since the last time this bit was cleared.                                                           |
|         |          | 1 Interrupt Detected<br>The slave has detected an address match, and a transfer has been initiated.                                                                                       |

## Interrupt Mask Register

The `TWI_IMSK` enables interrupt sources to assert the interrupt output. Each mask bit corresponds with one interrupt source bit in `TWI_ISTAT`. Reading and writing `TWI_IMSK` does not affect the contents of the `TWI_ISTAT`.

**TWI\_IMSK: Interrupt Mask Register - R/W**

Reset = 0x0000

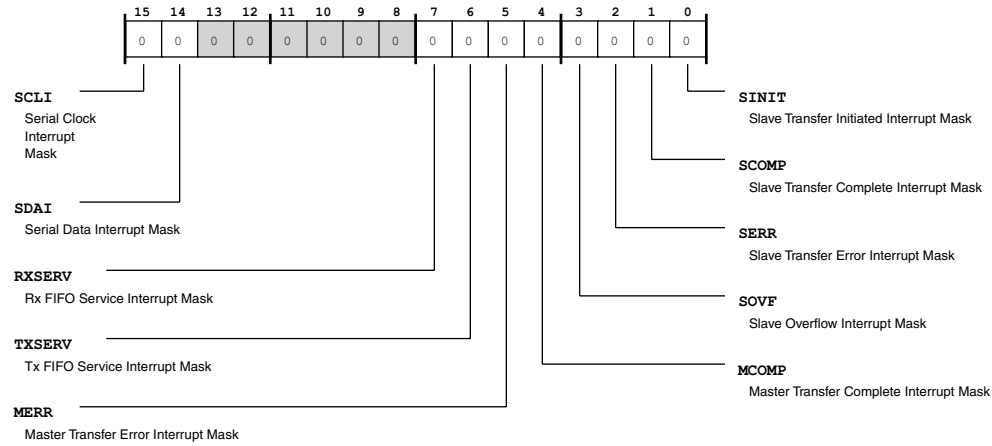


**Figure 20-20:** TWI\_IMSK Register Diagram

**Table 20-16:** TWI\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 15      | SCLI     | Serial Clock Interrupt Mask.    |
|         |          | 0 Mask (Disable) Interrupt      |
|         |          | 1 Unmask (Enable) Interrupt     |
| 14      | SDAI     | Serial Data Interrupt Mask.     |
|         |          | 0 Mask (Disable) Interrupt      |
|         |          | 1 Unmask (Enable) Interrupt     |
| 7       | RXSERV   | Rx FIFO Service Interrupt Mask. |
|         |          | 0 Mask (Disable) Interrupt      |
|         |          | 1 Unmask (Enable) Interrupt     |
| 6       | TXSERV   | Tx FIFO Service Interrupt Mask. |
|         |          | 0 Mask (Disable) Interrupt      |
|         |          | 1 Unmask (Enable) Interrupt     |

**Table 20-16:** TWI\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                  |                           |
|---------|----------|------------------------------------------|---------------------------|
| 5       | MERR     | Master Transfer Error Interrupt Mask.    |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |
| 4       | MCOMP    | Master Transfer Complete Interrupt Mask. |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |
| 3       | SOVF     | Slave Overflow Interrupt Mask.           |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |
| 2       | SERR     | Slave Transfer Error Interrupt Mask.     |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |
| 1       | SCOMP    | Slave Transfer Complete Interrupt Mask.  |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |
| 0       | SINIT    | Slave Transfer Initiated Interrupt Mask. |                           |
|         |          | 0                                        | Mask (Disable) Interrupt  |
|         |          | 1                                        | Unmask (Enable) Interrupt |

## FIFO Control Register

The `TWI_FIFCTL` control bits affect only the FIFO and are not tied in any way with master or slave mode operation.

**TWI\_FIFCTL: FIFO Control Register - R/W**

Reset = 0x0000

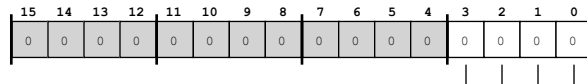


**Figure 20-21:** TWI\_FIFCTL Register Diagram

**Table 20-17:** TWI\_FIFCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | RXILEN   | Rx Buffer Interrupt Length.<br>The <code>TWI_FIFCTL.RXILEN</code> determines the rate at which receive buffer interrupts are to be generated. Interrupts may be generated with each byte received or after two bytes are received. Interrupt status is available in <code>TWI_FIFSTAT.RXSTAT</code> .                                                    |
|         |          | 0      RXSERVI on 1 or 2 Bytes in FIFO                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      RXSERVI on 2 Bytes in FIFO                                                                                                                                                                                                                                                                                                                        |
| 2       | TXILEN   | Tx Buffer Interrupt Length.<br>The <code>TWI_FIFCTL.TXILEN</code> determines the rate at which transmit buffer interrupts are to be generated. Interrupts may be generated with each byte transmitted or after two bytes are transmitted. Interrupt status is available in <code>TWI_FIFSTAT.TXSTAT</code> .                                             |
|         |          | 0      TXSERVI on 1 Byte of FIFO Empty                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      TXSERVI on 2 Bytes of FIFO Empty                                                                                                                                                                                                                                                                                                                  |
| 1       | RXFLUSH  | Rx Buffer Flush.<br>The <code>TWI_FIFCTL.RXFLUSH</code> directs the TWI to flush the contents of the receive buffer and update <code>TWI_FIFSTAT.RXSTAT</code> to indicate the buffer is empty. This state is held until this bit is cleared. During an active receive, the receive buffer in this state responds to the receive logic as if it is full. |
|         |          | 0      Normal Operation of Rx Buffer                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Flush Rx Buffer                                                                                                                                                                                                                                                                                                                                   |

Table 20-17: TWI\_FIFCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | TXFLUSH  | Tx Buffer Flush.<br>The TWI_FIFCTL.TXFLUSH directs the TWI to flush the contents of the transmit buffer and update TWI_FIFOSTAT.TXSTAT to indicate the buffer is empty. This state is held until this bit is cleared. During an active transmit, the transmit buffer in this state responds to the transmit logic as if it is empty. |
|         |          | 0 Normal Operation of Tx Buffer                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Flush Tx Buffer                                                                                                                                                                                                                                                                                                                    |

FIFO Status Register

The TWI\_FIFOSTAT fields indicate the state of the FIFO buffers' receive and transmit contents. The FIFO buffers do not discriminate between master data and slave data. By using the status and control bits provided, the FIFO can be managed to allow simultaneous master and slave operation.

TWI\_FIFOSTAT: FIFO Status Register - R/NW

Reset = 0x0000

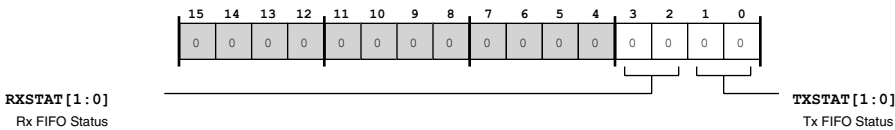


Figure 20-22: TWI\_FIFOSTAT Register Diagram



**Table 20-18:** TWI\_FIFOSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                 |                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| 3:2     | RXSTAT   | Rx FIFO Status.<br>The read-only <code>TWI_FIFOSTAT.RXSTAT</code> indicates the number of valid data bytes in the receive FIFO buffer. The status is updated with each FIFO buffer read using the peripheral data bus or write access by the receive shift register. Simultaneous accesses are allowed. |                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                       | Empty<br>The FIFO is empty.                                                                                                     |
|         |          | 1                                                                                                                                                                                                                                                                                                       | Contains 1 Byte<br>The FIFO contains one byte of data. A single byte peripheral read of the FIFO is allowed.                    |
|         |          | 2                                                                                                                                                                                                                                                                                                       | Reserved                                                                                                                        |
|         |          | 3                                                                                                                                                                                                                                                                                                       | Full<br>The FIFO is full and contains two bytes of data. Either a single or double byte peripheral read of the FIFO is allowed. |
| 1:0     | TXSTAT   | Tx FIFO Status.<br>The read-only <code>TWI_FIFOSTAT.TXSTAT</code> field indicates the number of valid data bytes in the FIFO buffer. The status is updated with each FIFO buffer write using the peripheral data bus or read access by the transmit shift register. Simultaneous accesses are allowed.  |                                                                                                                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                       | Empty<br>The FIFO is empty. Either a single or double byte peripheral write of the FIFO is allowed.                             |
|         |          | 1                                                                                                                                                                                                                                                                                                       | Contains 1 Byte<br>The FIFO contains one byte of data. A single byte peripheral write of the FIFO is allowed.                   |
|         |          | 2                                                                                                                                                                                                                                                                                                       | Reserved                                                                                                                        |
|         |          | 3                                                                                                                                                                                                                                                                                                       | Full<br>The FIFO is full and contains two bytes of data.                                                                        |

## Tx Data Single-Byte Register

The `TWI_TXDATA8` holds an 8-bit data value written into the FIFO buffer. Transmit data is entered into the corresponding transmit buffer in a first-in first-out order. For 16-bit peripheral bus writes, a write access to `TWI_TXDATA8` adds only one transmit data byte to the FIFO buffer. With each access, the transmit status (`TWI_FIFOSTAT.TXSTAT`) field is updated. If an access is performed while the FIFO buffer is full, the write is ignored and the existing FIFO buffer data and its status remains unchanged.

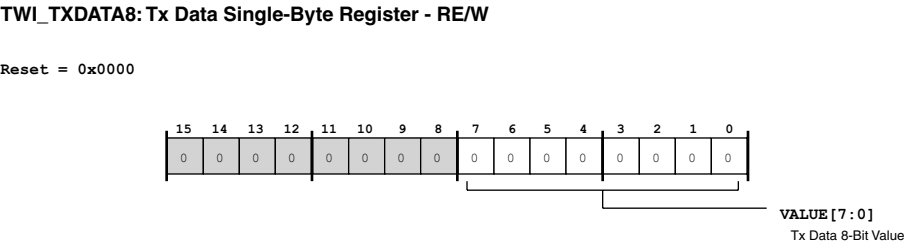


Figure 20-23: TWI\_TXDATA8 Register Diagram

Table 20-19: TWI\_TXDATA8 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | Tx Data 8-Bit Value.    |

## Tx Data Double-Byte Register

The `TWI_TXDATA16` holds a 16-bit data value written into the FIFO buffer. To reduce interrupt output rates and peripheral bus access times, a double byte transfer data access can be done. Two data bytes can be written, effectively filling the transmit FIFO buffer with a single access.

The data is written in little endian byte order, where byte 0 is the first byte to be transferred and byte 1 is the second byte to be transferred. With each access, the transmit status (`TWI_FIFOSTAT.TXSTAT`) field is updated. If an access is performed while the FIFO buffer is not empty, the write is ignored and the existing FIFO buffer data and its status remains unchanged.

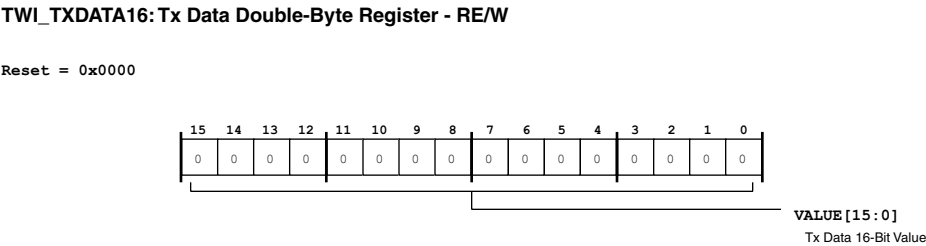


Figure 20-24: TWI\_TXDATA16 Register Diagram

**Table 20-20:** TWI\_TXDATA16 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Tx Data 16-Bit Value.   |

## Rx Data Single-Byte Register

The `TWI_RXDATA8` holds an 8-bit data value read from the FIFO buffer. Receive data is read from the corresponding receive buffer in a first-in first-out order. Although peripheral bus reads are 16 bits, a read access to `TWI_RXDATA8` accesses only one transmit data byte from the FIFO buffer. With each access, the receive status (`TWI_FIFOSTAT.RXSTAT`) field is updated. If an access is performed while the FIFO buffer is empty, the data is unknown and the FIFO buffer status remains indicating it is empty.

**TWI\_RXDATA8: Rx Data Single-Byte Register - RE/W**

Reset = 0x0000

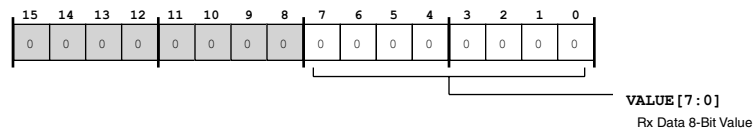


**Figure 20-25:** TWI\_RXDATA8 Register Diagram

**Table 20-21:** TWI\_RXDATA8 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | Rx Data 8-Bit Value.    |

## Rx Data Double-Byte Register

The `TWI_RXDATA16` holds a 16-bit data value read from the FIFO buffer. To reduce interrupt output rates and peripheral bus access times, a double byte receive data access can be performed. Two data bytes can be read, effectively emptying the receive FIFO buffer with a single access.

The data is read in little endian byte order, where byte 0 is the first byte received and byte 1 is the second byte received. With each access, the receive status (`TWI_FIFOSTAT.RXSTAT`) field is updated to indicate it is empty. If an access is performed while the FIFO buffer is not full, the read data is unknown and the existing FIFO buffer data and its status remains unchanged.

**TWI\_RXDATA16: Rx Data Double-Byte Register - RE/W**

Reset = 0x0000

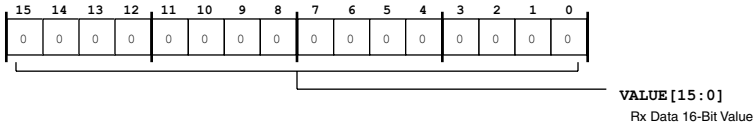


**Figure 20-26:** TWI\_RXDATA16 Register Diagram

**Table 20-22:** TWI\_RXDATA16 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Rx Data 16-Bit Value.   |

# 21 Controller Area Network (CAN)

The processor includes a controller area network (CAN) module, which implements the CAN 2.0B (active) protocol. This protocol is an asynchronous communications protocol used in both industrial and automotive control systems. The CAN protocol is well suited for control applications due to its capability to communicate reliably over a network. This is because the protocol incorporates CRC checking, message error tracking, and fault node confinement.

**NOTE:** This document assumes reader familiarity with the CAN standard. For more information, refer to Version 2.0 of the CAN Specification from Robert Bosch GmbH.

## CAN Features

The processor features one CAN module.

Key features of the CAN module include:

- Conformity to the CAN 2.0B (active) standard
- Dedicated acceptance mask for each mailbox
- Support for data rates of up to 1Mbit/s
- Support for standard (11-bit) and extended (29-bit) identifiers
- 32 mailboxes (8 transmit, 8 receive, 16 configurable)
- Data filtering (first 2 bytes) can be used for acceptance filtering (DeviceNet™ mode)
- Error status and warning registers
- Universal counter module
- Readable receive and transmit pin values
- Support for remote frames
- Active or passive network support
- Interrupts, including TX/RX complete, Error, and Global
- Clock derived from SCLK through a programmable divider, eliminating the need for an additional crystal

## CAN Functional Description

Following is a discussion of the functional operation of the CAN module.

### CAN Register List

The controller area network (CAN) module implements the CAN 2.0B (active) protocol. This protocol is an asynchronous communications protocol used in both industrial and automotive control systems. A set of registers govern CAN operations. For more information on CAN functionality, see the CAN register descriptions.

**Table 21-1:** CAN Module Registers

| Name       | Description                                              |
|------------|----------------------------------------------------------|
| CAN_MC1    | Mailbox Configuration 1 Register                         |
| CAN_MD1    | Mailbox Direction 1 Register                             |
| CAN_TRS1   | Transmission Request Set 1 Register                      |
| CAN_TRR1   | Transmission Request Reset 1 Register                    |
| CAN_TA1    | Transmission Acknowledge 1 Register                      |
| CAN_AA1    | Abort Acknowledge 1 Register                             |
| CAN_RMP1   | Receive Message Pending 1 Register                       |
| CAN_RML1   | Receive Message Lost 1 Register                          |
| CAN_MBTIF1 | Mailbox Transmit Interrupt Flag 1 Register               |
| CAN_MBRIF1 | Mailbox Receive Interrupt Flag 1 Register                |
| CAN_MBIM1  | Mailbox Interrupt Mask 1 Register                        |
| CAN_RFH1   | Remote Frame Handling 1 Register                         |
| CAN_OPSS1  | Overwrite Protection/Single Shot Transmission 1 Register |
| CAN_MC2    | Mailbox Configuration 2 Register                         |
| CAN_MD2    | Mailbox Direction 2 Register                             |
| CAN_TRS2   | Transmission Request Set 2 Register                      |

**Table 21-1:** CAN Module Registers (Continued)

| Name       | Description                                              |
|------------|----------------------------------------------------------|
| CAN_TRR2   | Transmission Request Reset 2 Register                    |
| CAN_TA2    | Transmission Acknowledge 2 Register                      |
| CAN_AA2    | Abort Acknowledge 2 Register                             |
| CAN_RMP2   | Receive Message Pending 2 Register                       |
| CAN_RML2   | Receive Message Lost 2 Register                          |
| CAN_MBTIF2 | Mailbox Transmit Interrupt Flag 2 Register               |
| CAN_MBRIF2 | Mailbox Receive Interrupt Flag 2 Register                |
| CAN_MBIM2  | Mailbox Interrupt Mask 2 Register                        |
| CAN_RFH2   | Remote Frame Handling 2 Register                         |
| CAN_OPSS2  | Overwrite Protection/Single Shot Transmission 2 Register |
| CAN_CLK    | Clock Register                                           |
| CAN_TIMING | Timing Register                                          |
| CAN_DBG    | Debug Register                                           |
| CAN_STAT   | Status Register                                          |
| CAN_CEC    | Error Counter Register                                   |
| CAN_GIS    | Global CAN Interrupt Status Register                     |
| CAN_GIM    | Global CAN Interrupt Mask Register                       |
| CAN_GIF    | Global CAN Interrupt Flag Register                       |
| CAN_CTL    | CAN Master Control Register                              |
| CAN_INT    | Interrupt Pending Register                               |
| CAN_MBTD   | Temporary Mailbox Disable Register                       |
| CAN_EWR    | Error Counter Warning Level Register                     |

**Table 21-1:** CAN Module Registers (Continued)

| Name               | Description                                   |
|--------------------|-----------------------------------------------|
| CAN_ESR            | Error Status Register                         |
| CAN_UCCNT          | Universal Counter Register                    |
| CAN_UCRC           | Universal Counter Reload/Capture Register     |
| CAN_UCCNF          | Universal Counter Configuration Mode Register |
| CAN_AMnnL          | Acceptance Mask (L) Register                  |
| CAN_AMnnH          | Acceptance Mask (H) Register                  |
| CAN_MBnn_DATA0     | Mailbox Word 0 Register                       |
| CAN_MBnn_DATA1     | Mailbox Word 1 Register                       |
| CAN_MBnn_DATA2     | Mailbox Word 2 Register                       |
| CAN_MBnn_DATA3     | Mailbox Word 3 Register                       |
| CAN_MBnn_LENGTH    | Mailbox Length Register                       |
| CAN_MBnn_TIMESTAMP | Mailbox Timestamp Register                    |
| CAN_MBnn_ID0       | Mailbox ID 0 Register                         |
| CAN_MBnn_ID1       | Mailbox ID 1 Register                         |

## ADSP-BF60x CAN Interrupt List

**Table 21-2:** ADSP-BF60x CAN Interrupt List

| Description   | ID | DMA Channel | Sensitivity |
|---------------|----|-------------|-------------|
| CAN0 Receive  | 40 |             | LEVEL       |
| CAN0 Transmit | 41 |             | LEVEL       |
| CAN0 Status   | 42 |             | LEVEL       |



## CAN External Interface

The interface to the CAN bus is a simple two-wire line. See the figures below for a symbolic representation of the CAN transceiver interconnection, as well as for a representative block diagram. Typically, the processor's CAN\_TX output and CAN\_RX input pins are connected to an external CAN transceiver's TX and RX pins (respectively). The CAN\_TX and CAN\_RX pins operate with TTL levels and are appropriate for operation with CAN bus transceivers according to ISO/DIS 11898.

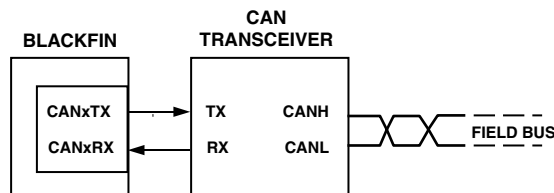


**Figure 21-1:** Representation of CAN Transceiver Interconnection

CAN data is defined to be either dominant (logic 0) or recessive (logic 1). The default state of the CAN\_TX output is recessive.

## ADSP-BF60x Specific External Interface

The CAN\_TX and CAN\_RX signals can be found on GPIO Port G, pins PG1 and PG4 respectively. By default, these pins are in GPIO mode. SPORT\_2A and the timer signals are also multiplexed on this pin. To enable CAN functionality, the appropriate bits must be set in the PORTG\_FER and PORTG\_MUX registers.

The PG4 pin (CAN\_RX input pin) is also internally routed to the alternate capture input TACI2 of GP timer 2. This way, GP timer 2 can be used to auto-detect or adjust the bit rate on the CAN bus.

**NOTE:** The CAN Pad does not support 5V operation.

## CAN Architectural Concepts

The full-CAN controller features 32 message buffers, which are called mailboxes. Eight mailboxes are dedicated for message transmission, eight are for reception, and 16 are programmable in direction.

The CAN module architecture is based around a 32-entry mailbox RAM. The mailbox is accessed sequentially by the CAN serial interface or the Blackfin core. Each mailbox consists of eight 16-bit control and data registers and two optional 16-bit acceptance mask registers, all of which must be configured before the mailbox itself is enabled.

Since the mailbox area is implemented as RAM, the reset values of these registers are undefined. The data is divided into fields, which includes a message identifier, a time stamp, a byte count, up to 8 bytes of data, and several control bits

|       |                                       |                     |
|-------|---------------------------------------|---------------------|
| WORD0 | BYTE 6   BYTE 7                       | CANx_MB00_DATA0     |
| WORD1 | BYTE 4   BYTE 5                       | CANx_MB00_DATA1     |
| WORD2 | BYTE 2   BYTE 3                       | CANx_MB00_DATA2     |
| WORD3 | BYTE 0   BYTE 1                       | CANx_MB00_DATA3     |
| WORD4 | DLC                                   | CANx_MB00_LENGTH    |
| WORD5 | TSV                                   | CANx_MB00_TIMESTAMP |
| WORD6 | EXTID_HI / DFC                        | CANx_MB00_ID0       |
| WORD7 | AME   RTR   IDE   BASEID   EXTID_LO   | CANx_MB00_ID1       |
| WORD8 | EXTID_HI / DFC                        | CANx_AM00L          |
| WORD9 | FDF   FMD   AMIDE   BASEID   EXTID_LO | CANx_AM00H          |

The CAN Mailbox Identification (CAN\_MBxx\_ID0/1) register pair includes:

- The 29 bit Identifier (base part BASEID plus extended part EXTID\_LO/HI)
- The Acceptance Mask Enable bit (AME)
- The Remote Transmission Request bit (RTR)
- The Identifier Extension bit (IDE)

**NOTE:** Do not write to the Identifier of a message object while the mailbox is enabled for the CAN module (the corresponding bit in CAN\_MCx is set).

The other mailbox area registers are:

- The Data Length Code (DLC) in CAN\_MBxx\_LENGTH. The upper 12 bits of this register of each mailbox are marked as reserved. These 12 bits should always be set to 0.
- Up to eight bytes for the Data field, sent MSB first from the CAN\_MBxx\_DATA3/2/1/0 registers, respectively, based on the number of bytes defined in the DLC. For example, if only one byte is transmitted or received (DLC = 1), then it is stored in the most significant byte of the CAN\_MBxx\_DATA3 register.
- Two bytes for the Time Stamp Value (TSV) in the CAN\_MBxx\_TIMESTAMP register

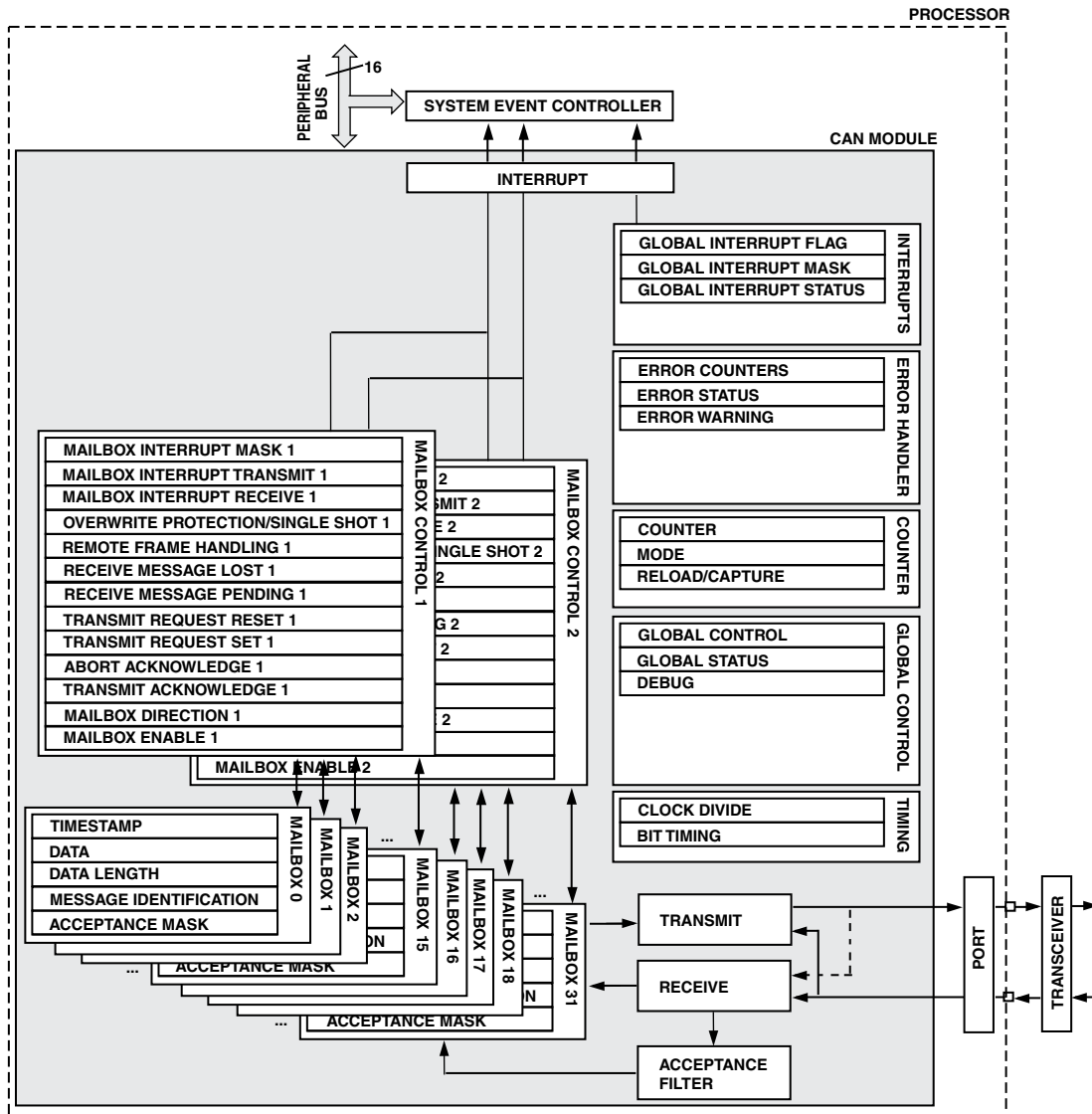
The final registers in the mailbox area are the Acceptance Mask Registers (CAN\_AMxxH and CAN\_AMxxL). The Acceptance Mask is enabled when the AME bit is set in the CAN\_MBxx\_ID1 register.

The “filtering on data field” option can be enabled by setting DNM bit in the CAN Master Control register (CAN\_CTL) and FDF bit in the corresponding Acceptance Mask register (CAN\_AMxxH). When enabled, EXTID\_HI[15:0] bits of CAN\_MBxx\_ID0 are reused as Acceptance Code (DFC) for the Data Field Filtering.

## CAN Block Diagram

The following figure shows a block diagram of the CAN module.

CAN Controller Block Diagram



## CAN Mailbox Control

Mailbox control MMRs function as control and status registers for the 32 mailboxes. Each bit in these registers represents one specific mailbox. Since CAN MMRs are all 16 bits wide, pairs of registers are required to manage certain functionality for all 32 individual mailboxes. Mailboxes 0-15 are configured/monitored in registers with a suffix of 1. Similarly, mailboxes 16-31 use the same named register with a suffix of 2. For example, the CAN mailbox direction registers (`CAN_MDx`) would control mailboxes as shown in the figure below.

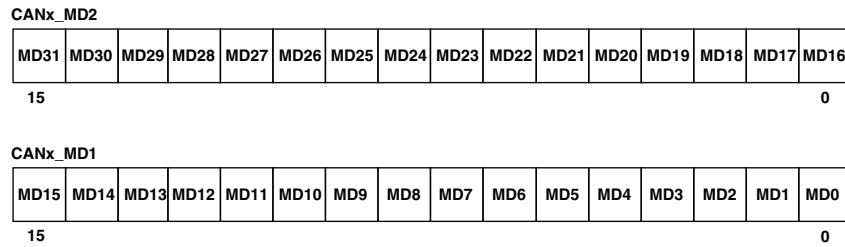


**Figure 21-2: CAN Mailbox Register Pair**

The Mailbox Control Register area consists of these register pairs:

- CAN\_MC1 and CAN\_MC2 (Mailbox Enable registers)
- CAN\_MD1 and CAN\_MD2 (Mailbox Direction registers)
- CAN\_TA1 and CAN\_TA2 (Transmit Acknowledge registers)
- CAN\_AA1 and CAN\_AA2 (Abort Acknowledge registers)
- CAN\_TRS1 and CAN\_TRS2 (Transmit Request Set registers)
- CAN\_TRR1 and CAN\_TRR2 (Transmit Request Reset registers)
- CAN\_RMP1 and CAN\_RMP2 (Receive Message Pending registers)
- CAN\_RML1 and CAN\_RML2 (Receive Message Lost registers)
- CAN\_RFH1 and CAN\_RFH2 (Remote Frame Handling registers)
- CAN\_OPSS1 and CAN\_OPSS2 (Overwrite Protection/Single Shot transmission registers)
- CAN\_MBIM1 and CAN\_MBIM2 (Mailbox Interrupt Mask registers)
- CAN\_MBTIF1 and CAN\_MBTIF2 (Mailbox Transmit Interrupt Flag registers)
- CAN\_MBRIF1 and CAN\_MBRIF2 (Mailbox Receive Interrupt Flag registers)

Since mailboxes 24–31 support transmit operation only and mailboxes 0–7 are receive-only mailboxes, the lower eight bits in the ‘1’ registers and the upper eight bits in the ‘2’ registers are sometimes reserved or are restricted in their usage.

## CAN Protocol Fundamentals

Although the CAN\_RX and CAN\_TX pins are TTL-compliant signals, the CAN signals beyond the transceiver have asymmetric drivers. A low state on the CAN\_TX pin activates strong drivers while a high state is driven weakly. Consequently, active low is called the *Dominant* state and active high is the *Recessive* state. If the CAN module is passive, the CAN\_TX pin is always high. If two CAN nodes transmit at the same time, dominant bits overwrite recessive bits.

The CAN protocol defines that all nodes trying to send a message on the CAN bus attempt to send a frame once the CAN bus becomes available. The start of frame indicator (SOF) signals the beginning of a new frame. Each CAN node then begins transmitting its message starting with the message ID.

While transmitting, the CAN controller samples the CAN\_RX pin to verify that the logic level being driven is the value it just placed on the CAN\_TX pin. This is where the names for the logic levels apply. If a transmitting node places a recessive 1 on the CAN\_TX pin and detects a dominant 0 on the CAN\_RX pin, it knows that another node has placed a dominant bit on the bus, which means another node is a higher priority.

Therefore, if the value sensed on the CAN\_RX pin is the value driven on the CAN\_TX pin, transmission continues, otherwise the CAN controller senses that it has lost arbitration and module configuration determines the next course of action.

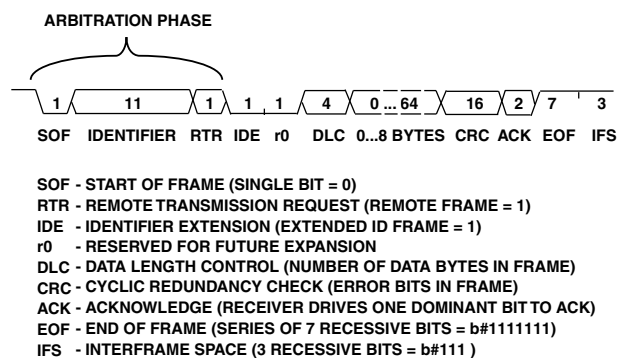


**Figure 21-3:** Standard CAN Frame

The figure above shows a basic 11-bit identifier frame. After the SOF and identifier is the RTR bit, which indicates whether the frame contains data (data frame) or is a request for data associated with the message identifier in the frame being sent (remote frame).

**NOTE:** In the CAN protocol, a dominant bit in the RTR field wins arbitration against a remote frame request (RTR = 1) for the same message ID, this allows a remote request to be a lower priority than a data frame.

The next field of interest is the IDE. When set, it indicates that the message is an extended frame with a 29-bit identifier instead of an 11-bit identifier. In an extended frame, the first part of the message resembles the figure below.

Therefore a dominant bit in the IDE field wins arbitration against an extended frame with the same lower 11-bits and standard frames are higher priority than extended frames.

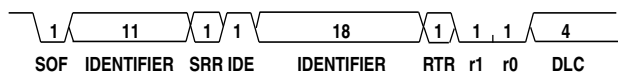


**Figure 21-4:** Extended CAN Frame

The substitute remote request bit (SRR, always sent as recessive), the reserved bits r0 and r1 (always sent as dominant), and the checksum (CRC) are generated automatically by the internal logic.

## CAN Data Transfer Modes

The following sections provide a discussion of the data transfer modes supported by the CAN controller.

### CAN Transmit Operation

The figure below shows the CAN transmit operation. Mailboxes 24-31 are dedicated transmitters. Mailboxes 8-23 can be configured as transmitters by writing 0 to the corresponding bit in the `CAN_MDx` register. After writing the data and the identifier into the mailbox area, the message is sent after mailbox *n* is enabled (`CAN_MCx.MBn = 1`) and, subsequently, the corresponding Transmit Request bit is set (`CAN_TRSx.MBn = 1`).

When a transmission completes, the corresponding bits in the `CAN_TRSx` and `CAN_TRRx` registers are cleared. If transmission was successful, the corresponding bit in the `CAN_TAx` register is set. If the transmission was aborted due to lost arbitration or a CAN error, the corresponding bit in the `CAN_AAx` register is set. A requested transmission can also be manually aborted by setting the corresponding bit in the `CAN_TRRx` register.

Multiple `CAN_TRSx.MBn` bits can be set simultaneously by software, and these bits are reset after either a successful or an aborted transmission.

These bits are also set by the CAN hardware in the following cases:

- when using the Auto-Transmit mode of the universal counter,
- when a message loses arbitration and the Single-Shot `CAN_OPSSx.MBn` bit is not set, or
- in the event of a remote frame request.

The latter is only possible for receive/transmit mail-boxes if the Automatic Remote Frame Handling feature is enabled (`CAN_RFHx.MBn = 1`).

Special care should be given to mailbox area management when a `CAN_TRSx` bit is set. Write access to the mailbox is permissible with a bit set, but changing data in such a mailbox may lead to unexpected data during transmission.

Enabling and disabling mailboxes has an impact on transmit requests. Setting the `CAN_TRSx` bit associated with a disabled mailbox may result in erroneous behavior. Similarly, disabling a mailbox before the associated `CAN_TRSx` bit is reset by the internal logic can cause unpredictable results.

### Retransmission

Normally, the current message object is sent again after arbitration is lost or an error frame is detected on the CAN bus line. If there is more than one transmit message object pending, the message object with the highest mailbox is sent first (see figure below). The currently aborted transmission is restarted after any messages with higher priority are sent.

A message which is currently under preparation is not replaced by another message which is written into the mailbox. The message under preparation is one that is copied into the temporary transmit buffer when the internal transmit request for the CAN core module is set. The message in the buffer is not replaced until it is sent successfully, the arbitration on the CAN bus line is lost, or there is an error frame on the CAN bus line.

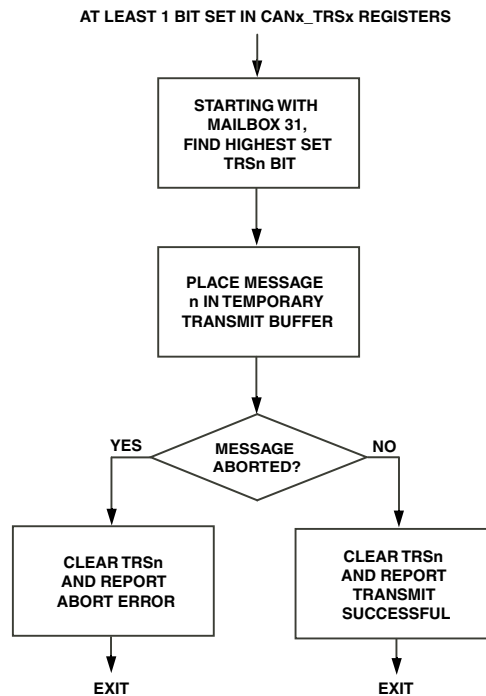


**Figure 21-5:** CAN Transmit Operation Flow Chart

### Single-Shot Transmission

If the Single Shot Transmission feature is used ( $CAN\_OPSSx.MBn = 1$ ), the corresponding  $CAN\_TRSx$  bit is cleared after the message is successfully sent or even if the transmission is aborted due to a lost arbitration or an error frame on the CAN bus line. Thus, there is no further attempt to transmit the message again if the initial try failed, and the Abort error is reported ( $CAN\_AAx.MBn = 1$ ).

### Auto-Transmission

In Auto-Transmit mode, the message in mailbox 11 (MB11) can be sent periodically using the universal counter. This mode is often used to broadcast heartbeats to all CAN nodes. Accordingly, messages sent this way usually have high priority.

The period value is written to the  $CAN\_UCRC$  register. The Auto-Transmission mode is enabled by setting the  $CAN\_UCCNF.UCCNF$  field to 0x03. When enabled, the counter  $CAN\_UCCNT$  is loaded with the value in the  $CAN\_UCRC$  register. The counter decrements at the CAN bit clock rate down to 0 and is then reloaded from  $CAN\_UCRC$ . Each time the counter reaches a value of 0, the  $CAN\_TRS1.MB11$  bit is automatically set by internal logic, and the corresponding message from mailbox 11 is sent.

For proper auto-transmit operation, mailbox 11 must be configured as a transmit mailbox and must contain valid data (identifier, control bits, and data) before the counter first expires after this mode is enabled.

**CAN Receive Operation**

The CAN hardware autonomously receives messages and discards invalid messages. Once a valid message is successfully received, the receive logic interrogates all enabled receive mailboxes sequentially, from mailbox 23 down to mailbox 0, whether the message is of interest to the local node or not.

Each incoming data frame is compared to all identifiers stored in active receive mailboxes (respective mailbox indices of CAN\_MD and CAN\_MC registers set to 1) and to all active transmit mailboxes with the Remote Frame Handling feature enabled (CAN\_RFHx.MBn = 1). The Message Identifier of the received message, along with the Identifier Extension (CAN\_MBxx\_ID1.IDE) and Remote Transmission Request (CAN\_MBxx\_ID1.RTR) bits, are compared against each mailbox’s register settings. In Standard mode, the message is compared to the content of the CAN\_MByy\_ID1 register. In extended mode, the content of the CANx\_MByy\_ID0 register must also match.

If the Acceptance Mask Enable (AME) bit in the CAN\_MByy\_ID1 register is not set, a match is signaled only if IDE, RTR, and all (11 or 29) identifier bits are exact. If, however, the AME is set, the Acceptance Mask Registers (CAN\_AMxxH/L) determine which of the IDE and RTR bits need to match.

The following logic applies:

$$[(\text{Received Message ID}) \text{ XNOR } (\text{CAN\_MBxx\_ID0/1})] \text{ OR } [(\text{AME}) \text{ AND } (\text{CAN\_AMxxH/L})].$$

This logic appears graphically in the figure below.

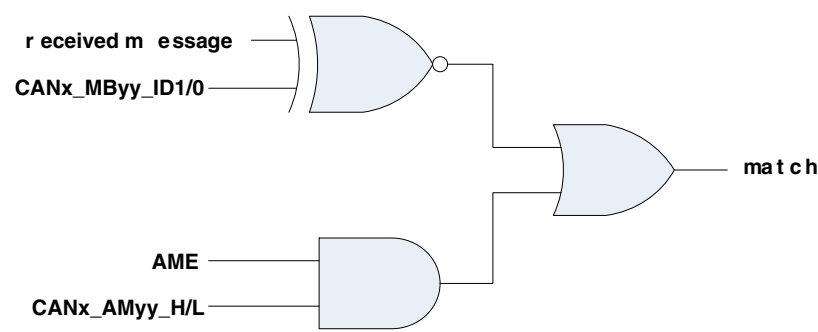


**Figure 21-6:** CAN Message Receive Logic

A one (1) at the respective bit position in the CAN\_AMxxH/L Mask registers means that the bit does not need to match when AME=1. This way, a mailbox can accept a group of messages.

**Table 21-3:** Mailbox Used for Acceptance Filtering

| MCn | MDn | RFHn | Mailbox n | Comment            |
|-----|-----|------|-----------|--------------------|
| 0   | x   | x    | Ignored   | Mailbox n disabled |



**Table 21-3:** Mailbox Used for Acceptance Filtering (Continued)

| MCn | MDn | RFHn | Mailbox n | Comment                                                                              |
|-----|-----|------|-----------|--------------------------------------------------------------------------------------|
| 1   | 0   | 0    | Ignored   | Mailbox n enabled, Mailbox n configured for transmit, Remote frame handling disabled |
| 1   | 0   | 1    | Used      | Mailbox n enabled, Mailbox n configured for transmit, Remote frame handling enabled  |
| 1   | 1   | x    | Used      | Mailbox n enabled, Mailbox n configured for receive                                  |

If the acceptance filter finds a matching identifier, the content of the received data frame is stored in that mailbox. A received message is stored only once, even if multiple receive mailboxes match its identifier. If the current identifier does not match any mailbox, the message is not stored.

The figure below illustrates the decision tree of the receive logic when processing the individual mailboxes.

If a message is received for a mailbox and that mailbox still contains unread data ( $RMP_n = 1$  in the Receive Message Pending register), then the user has to decide whether the old message should be overwritten or not. If the `CAN_OPSSx.OPSSn` bit is cleared, the corresponding `CAN_RMLx.RMLn` bit is set, and the stored message is overwritten. This results in the Receive Message Lost interrupt being raised (`CAN_GIS.RMLIS` is set). If, however, the `CAN_OPSSx.OPSSn` bit is set, the next mailboxes are checked for another matching identifier. If no match is found, the message is discarded, and the next message is checked.

*If a receive mailbox is disabled, an ongoing receive message for that mailbox is lost even if a second mailbox is configured to receive the same identifier.*

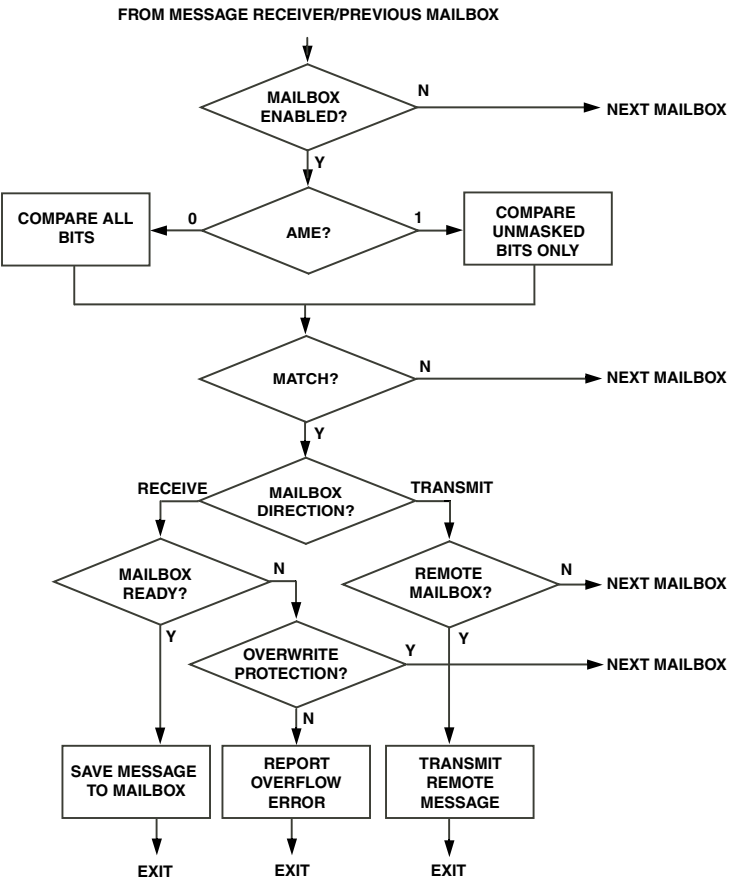


**Figure 21-7:** CAN Receive Operation Flow Chart

**CAN Data Acceptance Filtering**

If DeviceNet mode is enabled (`CAN_CTL.DNM = 1`) and the mailbox is set up for filtering on data field, the filtering is done on the standard ID of the message and data fields. The data field filtering can be programmed for either the first byte only or the first two bytes, as shown the table below.

If the `CAN_AMxxH.FDF` bit is set, the corresponding `CAN_AMxxL` register holds the Data Field Mask (DFM[15:0]). If the `FDF` bit is instead cleared, the corresponding `CAN_AMxxL` register holds the Extended Identifier Mask (EXTID\_HI[15:0]).

**Table 21-4:** Data Field Filtering

| FDF (Filter on Data Field) | FMD (Full Mask Data Field) | Description                              |
|----------------------------|----------------------------|------------------------------------------|
| 0                          | 0                          | Do not allow filtering on the data field |
| 0                          | 1                          | Not allowed. FMD must be 0 if FDF is 0   |

**Table 21-4:** Data Field Filtering (Continued)

| FDF (Filter on Data Field) | FMD (Full Mask Data Field) | Description                    |
|----------------------------|----------------------------|--------------------------------|
| 1                          | 0                          | Filter on first data byte only |
| 1                          | 1                          | Filter on first two data bytes |

## CAN Watchdog Mode

Watchdog mode is used to make sure that messages are received periodically. It is often used to observe whether or not a certain node on the network is alive and functioning properly, and, if not, to detect and manage its failure case accordingly.

This mode can be enabled by programming the Universal Counter to Watchdog mode by setting the `CAN_UCCNF.UCCNF[3:0]` to 0x2. Once enabled, the `CAN_UCCNT` register is loaded with the predefined value contained in `CAN_UCRC`. This counter then decrements at the CAN bit rate.

If the `CAN_UCCNF.UCCT` and `CAN_UCCNF.UCRC` bits are set and a message is received in mailbox 4 before the counter counts down to 0, the counter is reloaded with the `CAN_UCRC` contents. If the counter has counted down to 0 without receiving a message in mailbox 4, then the `CAN_GIS.UCEIS` bit is set, and the counter is automatically reloaded with the contents of the `CAN_UCRC` register. If an interrupt is desired for this event, the `CAN_GIM.UCEIM` bit must also be set. With the mask bit set, when a watchdog interrupt occurs, the `CAN_GIF.UCEIF` bit is also set.

The counter can be reloaded with the contents of `CAN_UCRC` or disabled by writing to the `CAN_UCCNF` register.

The time period it takes for the watchdog interrupt to occur is controlled by the value written into the `CAN_UCRC` register.

## CAN Time Stamps

To get an indication of the time of reception or the time of transmission for each message, program the CAN universal counter to Time Stamp Mode. This mode can be enabled by setting the `CAN_UCCNF.UCCNF[3:0]` field to 0x01.

If enabled, the value of the 16-bit free-running counter (`CAN_UCCNT`) is written into the `CAN_MBxx.TIMESTAMP` register of the corresponding mailbox when a received message is stored or a message is transmitted.

The Time Stamp Value is captured at the sample point of the Start Of Frame (SOF) bit of each incoming or outgoing message. Afterwards, this Time Stamp Value is copied to the `CAN_MBxx_TIMESTAMP` register of the corresponding mailbox.

If the mailbox is configured for Automatic Remote Frame Handling (`CAN_RFHx.RFHn = 1`), the Time Stamp Value is written for transmission of a Data frame (mailbox configured as transmit) or the reception of the requested Data frame (mailbox configured as receive).

The counter can be cleared by setting the `CAN_UCCNF.UCRC` bit to 1, or disabled by clearing the `CAN_UCCNF.UCE` bit. The counter can also be loaded with a value by writing to the counter register itself (`CAN_UCCNT`).

It is also possible to clear the counter (`CAN_UCCNT`) by reception of a message in mailbox number 4 (synchronization of all time stamp counters in the system). This is accomplished by setting the `CAN_UCCNF.UCCT` bit.

An overflow of the counter sets the `CAN_GIS.UCEIS` bit. A Global CAN Interrupt can optionally occur by unmasking the `CAN_GIM.UCEIM` bit. If the interrupt source is unmasked, the `CAN_GIF.UCEIF` bit is also set.

## **CAN Remote Frame Handling**

Automatic handling of remote frames can be enabled for a transmit mailbox by setting the corresponding `CAN_RFHx.RFHn` bit of a transmit mailbox.

Remote frames are Data frames with no data field and the RTR bit set. The Data Length Code (DLC) of the responding data frame is overruled by the DLC of the requesting Remote Frame. A DLC can be programmed with values in the range of 0 to 15, but DLC values greater than 8 are considered as 8.

A remote frame contains:

- the Identifier bits
- the control field DLC (Data Length Count)
- the Remote Transmission Request (RTR) bit

Only configurable mailboxes, MB8–MB23, can process remote frames, but all mailboxes can receive and transmit remote frame requests. When setup for Automatic Remote Frame Handling, the `CAN_OPSSx` register has no effect. All content of a mailbox is always overwritten by an incoming message.

*If a Remote frame is received, the DLC of the corresponding mailbox is overwritten with the received value.*

Erroneous behavior may result when the `RFHn` bit is changed while the corresponding mailbox is currently being processed. To avoid the risk of inconsistent messages, it is recommended to temporarily disable the mailbox while its Data registers are updated.

## CAN Mailbox Disabling (Temporary)

If a mailbox is enabled and configured as “transmit,” write accesses to the data field should be guarded to avoid transmission of inconsistent messages. Special care must be taken if the mailbox is transmitting (or attempting to transmit) repeatedly. Also, if this mailbox is used for Automatic Remote Frame Handling, the data field must be updated without losing an incoming Remote Request frame and without sending inconsistent data. Therefore, the CAN controller allows for temporary mailbox disabling, which can be enabled by programming the Mailbox Temporary Disable register (CAN\_MBTDR).

The pointer to the requested mailbox must be written to the CAN\_MBTDR.TDPTR[4:0] field, and the CAN\_MBTDR.TDR bit must be set. The corresponding CAN\_MBTDR.TDA flag is subsequently set by the internal logic.

If a mailbox is configured as “transmit” ( $MD_n = 0$ ) and TDA bit is set, the content of the Data Field of that mailbox can be updated. If there is an incoming remote Request Frame while the mailbox is temporarily disabled, the corresponding Transmit Request Set bit (TRS<sub>n</sub>) is set by the internal logic and the Data Length Code (DLC) of the incoming message is written to the corresponding mailbox. However, the message being requested is not sent until the CAN\_MBTDR.TDR bit is cleared. Similarly, all transmit requests for temporarily disabled mailboxes are ignored until the CAN\_MBTDR.TDR bit is cleared. Additionally, transmission of a message is immediately aborted if the mailbox is temporarily disabled and the corresponding Transmission Request Reset (TRR<sub>n</sub>) bit for this mailbox is set.

If a mailbox is configured as “receive” ( $MD_n = 1$ ), then after issuing a Temporary Disable Request, the CAN\_MBTDR.TDA flag is set, and the mailbox is not processed. If there is an incoming message for a mailbox being temporarily disabled, the internal logic waits until the reception is complete or there is an error on the CAN bus before setting CAN\_MBTDR.TDA. Once this flag is set, the mailbox can then be completely disabled ( $MC_n = 0$ ) without the risk of losing an incoming frame. The CAN\_MBTDR.TDR bit must then be reset as soon as possible.

When the CAN\_MBTDR.TDA flag is set for a given mailbox, only the Data field of that mailbox can be updated. Accesses to the Control bits and the Identifier are denied.

## CAN Operating Modes

The CAN controller is in configuration mode when coming out of processor reset or hibernate. It is only when the CAN is in configuration mode that hardware behavior can be altered. Before initializing the mailboxes themselves, the CAN bit timing must be set up to work on the CAN bus to which the controller is expected to connect.

## Bit Timing

The CAN controller does not have a dedicated clock. Instead, the CAN clock is derived from the system clock based on a configurable number of time quanta. The Time Quantum (TQ) is derived from the formula:

$$TQ = (BRP+1)/SCLK,$$

where BRP is the 10-bit Bit Rate Prescaler field in the CAN\_CLK register.

Although the BRP field can be set to any value, it is recommended that the value be greater than or equal to 4, as restrictions apply to the bit timing configuration when BRP is less than 4.

The CAN\_CLK register defines the TQ value, and multiple time quanta make up the duration of a CAN bit on the bus. The CAN\_TIMING register controls the nominal bit time and the sample point of the individual bits in the CAN protocol. The figure below shows the three phases of a CAN bit—the synchronization segment, the segment before the sample point, and the segment after the sample point.

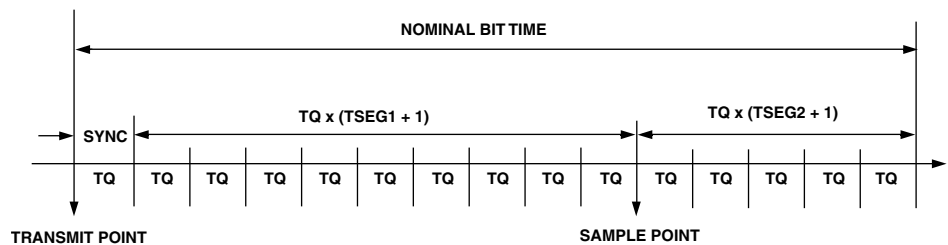


**Figure 21-8:** Three Phases of a CAN Bit

The synchronization segment is fixed to one TQ. It is required to synchronize the nodes on the bus. All signal edges are expected to occur within this segment.

The TSEG1 and TSEG2 fields of the CAN\_TIMING register control how many TQs the CAN bits consist of, resulting in the CAN bit rate. The nominal bit time is given by the formula,  $t_{BIT} = TQ \times [1 + (1 + TSEG1) + (1 + TSEG2)]$

For safe receive operation on given physical networks, the sample point is programmable by the TSEG1 field. The TSEG2 field holds the number of TQs needed to complete the bit time. Often, best sample reliability is achieved with sample points in the high 80% range of the bit time. Never use sample points lower than 50%. Thus, TSEG1 should always be greater than or equal to TSEG2.

The CAN module does not distinguish between the Propagation Segment and the Phase Segment-1 as defined by the standard. The TSEG1 value is intended to cover both of them. The TSEG2 value represents the Phase Segment-2.

If the CAN module detects a recessive-to-dominant edge outside the synchronization segment, it can automatically move the sampling point such that the CAN bit is still handled properly. The Synchronization Jump Width (SJW) field specifies the maximum number of TQs, ranging from 1 to 4 (SJW + 1), allowed for such a re-synchronization attempt. The SJW value should not exceed TSEG2 or TSEG1. Therefore, the fundamental rule for writing CAN\_TIMING is:

$SJW \leq TSEG2 \leq TSEG1$

In addition to this fundamental rule, Phase Segment-2 must also be greater than or equal to the Information Processing Time (IPT). This is the time required by the logic to sample the CAN\_RX input, which is 3 system clock cycles.

Because of this, restrictions apply to the minimal value of TSEG2 if the Bit Clock Prescaler (BRP) is lower than 2. If BRP is set to 0, the TSEG2 field must be greater than or equal to 2. Else if the BRP is set to 1, the minimum TSEG2 is 1.

**NOTE:** All nodes on a CAN bus should use the same nominal bit rate.

With all the timing parameters set, the final consideration is how sampling is performed. The default behavior of the CAN controller is to sample the CAN bit once at the sampling point described by the CAN\_TIMING register, controlled by the SAM bit. If the SAM bit is set, however, the input signal is oversampled three times at the system clock rate. The resulting value is generated by a majority decision of the three sample values. Always keep the SAM bit cleared if the BRP value is less than 4.

Do not modify the CAN\_CLK or CAN\_TIMING registers during normal operation. Always enter configuration mode first. Writes to these registers have no effect if not in configuration or debug mode. If not coming out of processor reset or hibernate, enter configuration mode by setting the CAN\_CTL.CCR bit and poll the CAN\_STAT register until CAN\_STAT.CCA is set.

**NOTE:** If the CAN\_TIMING.TSEG1 field is programmed to 0, the module doesn't leave the configuration mode.

During configuration mode, the module is not active on the CAN bus line. The CAN\_TX output pin remains recessive and the module does not receive/transmit messages or error frames. After leaving the configuration mode, all CAN core internal registers and the CAN error counters are set to their initial values.

A soft reset does not change the values of CAN\_CLK and CAN\_TIMING. Thus, an ongoing transfer through the CAN bus cannot be corrupted by changing the bit timing parameter or initiating the soft reset (by setting the CAN\_CTL.SRS bit).

## CAN Low Power Features

The CAN module includes built-in Sleep and Suspend modes to save power. It also responds to the Hibernate power state on processors that support that state. The behavior of the CAN module in these modes is described in the following sections.

### CAN Built-In Suspend Mode

The most modest of power savings mode is the suspend mode. This mode is entered by setting the CAN\_CTL.CSR bit. The module enters the suspend mode after the current operation of the CAN bus is finished, at which point the internal logic sets the CAN\_STAT.CSA bit. Once this mode is entered, the module is no longer active on the CAN bus line, slightly reducing power consumption. The CAN\_TX output

pin remains at recessive state, and the module does not receive/transmit messages or error frames. The content of the CAN Error Counter register remains unchanged. The Suspend mode can subsequently be exited by clearing `CAN_CTL.CSR`. The only differences between Suspend mode and Configuration mode are that writes to the `CAN_CLK` and `CAN_TIMING` registers are still locked in suspend mode, and the `CAN_CTL` and `CAN_STAT` registers are not reset when exiting suspend mode.

## **CAN Built-In Sleep Mode**

The next level of power savings can be realized by using the CAN module's built-in sleep mode. This mode is entered by setting the `CAN_CTL.SMR` bit. The module enters the sleep mode after the current operation of the CAN bus is finished. Once this mode is entered, many of the internal CAN module clocks are shut off, reducing power consumption, and the `CAN_INT.SMACK` bit is set. When the CAN module is in sleep mode, all register reads return the contents of `CAN_INT` instead of the usual contents. All register writes, except to `CAN_INT`, are ignored in sleep mode. A small part of the module is clocked continuously to allow for wakeup out of sleep mode. A write to the `CAN_INTR` register ends sleep mode. If the `CAN_CTL.WBA` bit is set before entering sleep mode, a dominant bit on the `CAN_RX` pin also ends sleep mode. When software sets the `CAN_CTL.SMR` bit, hardware sets the `CAN_CTL.CSR` bit as well, making Sleep mode a superset of Suspend mode. When the controller wakes up from Sleep mode, hardware automatically clears `CAN_CTL.SMR` and `CAN_CTL.CSR`. If, however, the controller never enters sleep mode, because the wake-up condition was met before `CAN_INT.SMACK` bit turns to one, the `CAN_CTL.SMR` and `CAN_CTL.CSR` may not get cleared automatically. Therefore, it is good programming practice to always clear those two bits by software when returning from sleep mode.

## **CAN Wakeup From Hibernate State**

Many processors provide a hibernate state, where the internal voltage regulator shuts off the internal power supply to the chip, turning off the core and system clocks in the process. In this mode, the only power drawn is that used by the regulator circuitry awaiting any of the possible hibernate wakeup events. One such event is a wakeup due to CAN bus activity. After hibernation, the CAN module must be re-initialized. For low power designs, the external CAN bus transceiver is typically put into standby mode through one of the processor's general purpose I/O pins. While in standby mode, the CAN transceiver continually drives the recessive logic '1' level onto the `CAN_RX` pin. If the transceiver then senses CAN bus activity, it will, in turn, drive the `CAN_RX` pin to the dominant logic '0' level. This signals to the processor that CAN bus activity is detected. If the internal voltage regulator is programmed to recognize CAN bus activity as an event to exit hibernate state, the part responds appropriately. Otherwise, the activity on the `CAN_RX` pin has no effect on the processor state.

## **Soft Reset**

The CAN controller features a build-in reset mechanism which is referred to as Soft Reset. Soft reset is entered immediately after software has set `CAN_CTL.SRS` bit. Soft reset brings all control registers to a defined state. Mailbox and error registers remain unaffected. Soft reset does not alter the `CAN_TIMING` and `CAN_CLK` registers and does not disturb the on-going transmission of a currently pending message, acknowledge bit or error frame. However, when recovering from soft reset, software may lose track of transmission or reception reports and interrupts.



## CAN Event Control

The following is a description of how CAN events are generated and controlled.

### CAN Interrupt Signals

The CAN module provides three independent interrupts: two mailbox interrupts (Mailbox Receive Interrupt (MBRIRQ) and mailbox transmit interrupt (MBTIRQ) and a global CAN status interrupt (GIRQ). The values of these three interrupts can also be read back through the CAN\_GIS registers.

### Mailbox Interrupts

Each of the 32 mailboxes in the CAN module may generate a receive or transmit interrupt, depending on the mailbox configuration. To enable a mailbox to generate an interrupt, set the corresponding CAN\_MBIMn bit.

If a mailbox is configured as a receive mailbox, the corresponding CAN\_MBRIFn bit and CAN\_RMPn bit are set after a received message is stored in mailbox n. If the automatic remote frame handling feature is used (CAN\_RFHn = 1), the Receive Interrupt Flag is set after the requested data frame is stored in the mailbox.

If any CAN\_MBRIF bits are set, the MBRIRQ interrupt output is raised in CAN\_INTR. In order to clear the MBRIRQ interrupt request, all of the set MBRIFn bits must be cleared by software by writing a 1 to those set bit locations in CAN\_MBRIFn. Prior to this, the corresponding CAN\_RMPn bit must also be cleared by software.

If a mailbox is configured as a transmit mailbox, the corresponding CAN\_MBTIFn bit in the transmit interrupt flag is set after the message in mailbox n is sent correctly, and the corresponding CAN\_TAn bit also gets set. The TAn bits maintain state even after the corresponding mailbox n is disabled (MCn = 0). If the automatic remote frame handling feature is used, then transmit interrupt flag is set after the requested data frame is sent from the mailbox.

If any MB bits are set in CAN\_MBTIFn, the MBTIRQ interrupt output is raised in CAN\_INTR. In order to clear the MBTIRQ interrupt request, all of the set MBTIFn bits must be cleared by software by writing a 1 to those set bit locations in CAN\_MBTIFn. Additionally, software must clear the associated TAn bit or set the associated TRSn bit to clear the interrupt source that asserts the MBTIFn bit.

### Global CAN Interrupt

The global CAN interrupt logic is implemented with three registers:

- The CAN\_GIM register, where each interrupt source can be enabled or disabled separately
- The CAN\_GIS register
- The CAN\_GIF register

The interrupt mask bits only affect the content of the `CAN_GIF` register. If the mask bit is not set in the `CAN_GIM` register, the corresponding flag bit is not set when the event occurs. The interrupt status bits in the `CAN_GIS` register, however, are always set if the corresponding interrupt event occurs, independent of the mask bits. Thus, the interrupt status bits can be used for polling of interrupt events.

The `CAN_INTR.GIRQ` bit is only asserted if a bit in the `CAN_GIF` register is set. The read-only `GIRQ` bit remains set as long as at least one bit in `CAN_GIF` is set. All bits in the Interrupt Status and Interrupt Flag registers remain set until cleared by software or a soft reset has occurred.

*The `CAN_GIF` register is read-only (RO). In the Global CAN Interrupt ISR, the interrupt latch should be cleared by writing to 1 to the corresponding bit of the `CAN_GIS` register, which clears the related bits of the `CAN_GIS` and `CAN_GIF` registers, as well as the `CAN_INTR.GIRQ` bit.*

There are several interrupt events that can activate this `GIRQ` interrupt:

- Access denied interrupt (ADIM, ADIS, ADIF): At least one access to the mailbox RAM occurred during a data update by internal logic.
- Universal counter exceeded interrupt (UCEIM, UCEIS, UCEIF): There was an overflow of the universal counter (in Time Stamp mode or Event Counter mode) or the counter has reached the value 0x0000 (in Watchdog mode).
- Receive message lost interrupt (RMLIM, RMLIS, RMLIF): A message is received for a mailbox that currently contains unread data. At least one bit in the `CAN_RMLn` register is set. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and there is at least one bit in `CAN_RMLn` still set, then the bit in `CAN_GIS` (and `CAN_GIF`) is not set again. The internal interrupt source signal is only active if a new bit in `CAN_RMLn` is set.
- Abort acknowledge interrupt (AAIM, AAIS, AAIF): At least one `AAn` bit in the `CAN_AAn` registers is set. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and there is at least one bit in `CAN_AAn` still set, then the bit in `CAN_GIS` (and `CAN_GIF`) is not set again. The internal interrupt source signal is only active if a new bit in `CAN_AAn` is set. The `AAn` bits maintain state even after the corresponding mailbox `n` is disabled (`MCn = 0`).
- Access to unimplemented address interrupt (UIAIM, UIAIS, UIAIF): There was a CPU access to an address which is not implemented in the controller module.
- Wakeup interrupt (WUIM, WUIS, WUIF): The CAN module has left the sleep mode because of detected activity on the CAN bus line.
- Bus-Off interrupt (BOIM, BOIS, BOIF): The CAN module has entered the bus-off state. This interrupt source is active if the status of the CAN core changes from normal operation mode to the bus-off mode. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and the bus-off mode is still active, then this bit is not set again. If the module leaves the bus-off mode, the bit in `CAN_GIS` (and `CAN_GIF`) remains set, if not explicitly cleared.
- Error-passive interrupt (EPIM, EPIS, EPIF): The CAN module has entered the error-passive state. This interrupt source is active if the status of the CAN module changes from the error-active mode to the error-passive mode. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and the error-passive mode is still

active, then this bit is not set again. If the module leaves the error-passive mode, the bit in `CAN_GIS` (and `CAN_GIF`) remains set, if not explicitly cleared.

- Error warning receive interrupt (EWRIM, EWRIS, EWRIF): The CAN receive error counter (RXCNT) has reached the warning limit. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and the error warning mode is still active, this bit is not set again. If the module leaves the error warning mode, the bit in `CAN_GIS` (and `CAN_GIF`) remains set, if not explicitly cleared.
- Error warning transmit interrupt (EWTIM, EWTIS, EWTIF): The CAN transmit error counter (TXCNT) has reached the warning limit. If the bit in `CAN_GIS` (and `CAN_GIF`) is cleared and the error warning mode is still active, this bit is not set again. If the module leaves the error warning mode, the bit in `CAN_GIS` (and `CAN_GIF`) remains set, if not explicitly cleared.

## CAN Event Counter

For diagnostic functions, it is possible to use the universal counter as an event counter. The counter can be programmed in the `CAN_UCCNF.UCCNF[3:0]` field to increment on one of these conditions:

- `UCCNF[3:0] = 0x6` – CAN error frame. Counter is incremented if there is an error frame on the CAN bus line.
- `UCCNF[3:0] = 0x7` – CAN overload frame. Counter is incremented if there is an overload frame on the CAN bus line.
- `UCCNF[3:0] = 0x8` – Lost arbitration. Counter is incremented every time arbitration on the CAN line is lost during transmission.
- `UCCNF[3:0] = 0x9` – Transmission aborted. Counter is incremented every time arbitration is lost or a transmit request is canceled (AAn is set).
- `UCCNF[3:0] = 0xA` – Transmission succeeded. Counter is incremented every time a message sends without detected errors (TAn is set).
- `UCCNF[3:0] = 0xB` – Receive message rejected. Counter is incremented every time a message is received without detected errors but not stored in a mailbox because there is no matching identifier found.
- `UCCNF[3:0] = 0xC` – Receive message lost. Counter is incremented every time a message is received without detected errors but not stored in a mailbox because the mailbox contains unread data (RMLn is set).
- `UCCNF[3:0] = 0xD` – Message received. Counter is incremented every time a message is received without detected errors, whether the received message is rejected or stored in a mailbox.
- `UCCNF[3:0] = 0xE` – Message stored. Counter is incremented every time a message is received without detected errors, has an identifier that matches an enabled receive mailbox, and is stored in the receive mailbox (RMPn is set).

- `UCCNF[3:0] = 0xF` – Valid message. Counter is incremented every time a valid transmit or receive message is detected on the CAN bus line.

## CAN Warnings and Errors

CAN warnings and errors are controlled using the Error Counter (`CAN_CEC`) register, the Error Status (`CAN_ESR`) register, and the Error Counter Warning Level (`CAN_EWR`) register.

### Programmable Warning Limits

It is possible to program the warning level for `EWTIS` and `EWRI` separately by writing to the `CAN_EWR.EWLREC` and `CAN_EWR.EWLTEC` fields. After power-on reset, the `CAN_EWR` register is set to the default warning level of 96 for both error counters. After soft reset, the contents of this register remain unchanged.

### Error Handling

Error management is an integral part of the CAN standard. Five different kinds of bus errors may occur during transmissions:

- Bit error – A bit error can be detected by the transmitting node only. Whenever a node is transmitting, it continuously monitors its receive pin (`CAN_RX`) and compares the received bit with the transmitted bit. During the arbitration phase, the node simply postpones the transmission if the received and transmitted bits do not match. However, after the arbitration phase (that is, once the `RTR` bit is sent successfully), a bit error is signaled any time the value on `CAN_RX` does not equal what is being transmitted on `CAN_TX`.
- Form error – A form error occurs any time a fixed-form bit position in the CAN frame contains one or more illegal bits-- that is, when a dominant bit is detected at a delimiter or end-of-frame bit position.
- Acknowledge error – An acknowledge error occurs whenever a message is sent and no receivers drive an acknowledge bit.
- CRC error – A CRC error occurs whenever a receiver calculates the CRC on the data it received and finds it different than the CRC that was transmitted on the bus itself.
- Stuff error – The CAN specification requires the transmitter to insert an extra stuff bit of opposite value after 5 bits have been transmitted with the same value. The receiver disregards the value of these stuff bits. However, it takes advantage of the signal edge to re-synchronize itself. A stuff error occurs on receiving nodes whenever the 6th consecutive bit value is the same as the previous five bits.

Once the CAN module detects any of the above errors, it updates the `CAN_ESR` and `CAN_CEC` registers. In addition to the standard errors, the `CAN_ESR` register features the `SA0` flag, which signals when the `CAN_RX` pin sticks at dominant level, indicating that shorted wires are likely.

## Error Frames

It is of central importance that all nodes on the CAN bus ignore data frames that one single node failed to receive. To accomplish this, every node sends an error frame as soon as it has detected an error. See the figure below.

Once a device has detected an error, it still completes the ongoing bit and initiates an Error Frame by sending six dominant and eight recessive bits to the bus. This is a violation to the bit stuffing rule and informs all nodes that the ongoing frame needs to be discarded.

All receivers that did not detect the transmission error in the first instance now detect a stuff bit error. The transmitter may detect a normal bit error sooner. It aborts the transmission of the ongoing frame and tries sending it again later.

Finally, all nodes on the bus have detected an error. Consequently, all of them send 6 dominant and 8 recessive bits to the bus as well. The resulting error frame consists of two different fields. The first field is given by the superposition of error flags contributed from the different stations, which is a sequence of 6 to 12 dominant bits. The second field is the error delimiter and consists of 8 recessive bits indicating the end of frame.

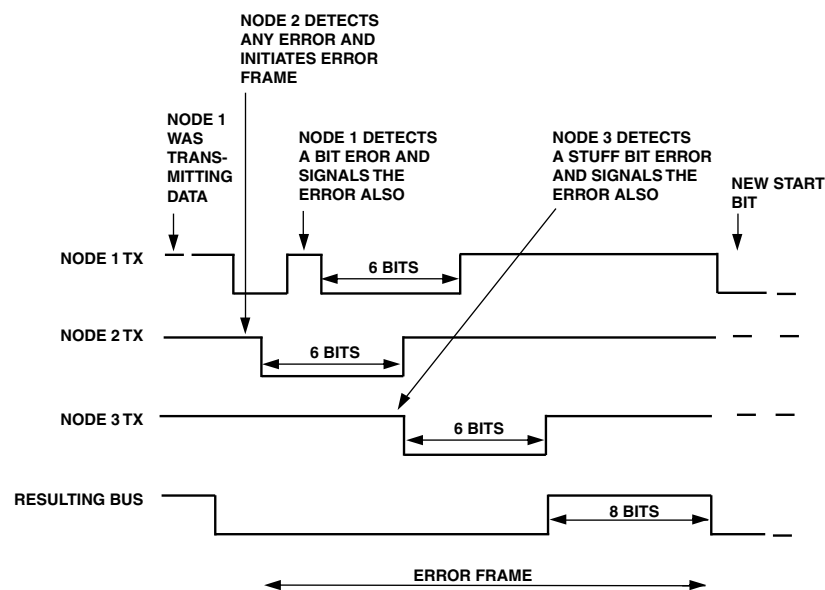


**Figure 21-9:** CAN Error Scenario Example

For CRC errors, the error frame is initiated at the end of the frame, rather than immediately after the failing bit.

After having received 8 recessive bits, every node knows that the error condition is resolved and starts transmission if messages are pending. The former transmitter that had to abort its operation must win the new arbitration again; otherwise its message is delayed as determined by priority.

Because the transmission of an error frame destroys the frame under transmission, a faulty node erroneously detecting an error can block the bus. Because of this, there are two node states which determine a

nodes right to signal an error—error-active and error-passive. Error-active nodes are those which have an error detection rate below a certain limit. These nodes drive an ‘Active Error Flag’ of 6 dominant bits.

Nodes with a higher error detection rate are suspected of having a local problem and, therefore, have a limited right to signal errors. These error-passive nodes drive a ‘Passive Error Flag’ consisting of 6 recessive bits. Thus, an error-passive transmitting node is still able to inform the other nodes about the abortion of a self-transmitted frame, but it is no longer able to destroy correctly received frames of other nodes.

## Error Levels

The CAN specification requires each node in the system to operate in one of three levels. See the table below. This prevents nodes with high error rates from blocking the entire network, as the errors might be caused by local hardware. The CAN module provides an error counter for transmit (TEC) and an error counter for receive (REC). The `CAN_CEC` register houses each of these 8-bit counters.

After initialization, both the TEC and the REC counters are 0. Each time a bus error occurs, one of the counters is incremented by either 1 or 8, depending on the error situation (documented in Version 2.0 of the CAN Specification). Successful transmit or receive operations decrement the respective counter by 1.

If either of the error counters exceeds 127, the CAN module goes into an error-passive state and the `CAN_STATUS.EP` bit is set. Once this happens, it is not allowed to send any more active error frames. However, it is still allowed to transmit messages and to signal passive error frames in case the transmission fails because of a bit error.

If one of the counters exceeds 255 (that is, when an 8-bit counter overflows), the CAN module is disconnected from the bus. It goes into Bus-Off mode and the `CAN_STAT.EBO` bit is set. Software intervention is required to recover from this state, unless the `CAN_CTL.ABO` bit is enabled, which allows entering of active mode after the Bus-Off recovery sequence.

**Table 21-5:** CAN Error Level Description

| Level         | Condition                                                              | Description                                                                                                                        |
|---------------|------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Error active  | Transmit and receive error counters <128                               | This is the initial condition level. As long as errors stay below 128, the node will drive active error flags during error frames. |
| Error passive | Transmit or receive error counter value between 128 and 255, inclusive | Errors have accumulated to a level that requires the node to drive passive error flags during error frames.                        |
| Bus off       | Transmit or receive error counters greater than 255                    | CAN module goes into bus-off mode                                                                                                  |

In addition to these levels, the CAN module also provides a warning mechanism, which is an enhancement to the CAN specification. There are separate warnings for transmit and receive. By default, when one of the error counters exceeds 96, a warning is signaled and is represented in the `CAN_STAT` register by either

the CAN Receive Warning Flag (WR) bit when RXECNT is at limit; or CAN Transmit Warning Flag (WT) bit when TXECNT is at limit. The error warning level can be programmed using the error warning register (CAN\_EWR).

Additionally, interrupts can occur for all of these levels by unmasking them in the global CAN interrupt mask register (CAN\_GIM). These interrupts include the Bus-Off Interrupt (BOIM), the Error-Passive Interrupt (EPIM), the error warning receive interrupt (EWRIM), and the Error Warning Transmit Interrupt (EWTIM).

During the bus-off recovery sequence, the configuration mode request (CCR) bit in the CAN master control (CAN\_CTL) register is set by the internal logic, thus the CAN core module does not automatically come out of the bus-off mode. The CCR bit cannot be reset until the bus-off recovery sequence is finished.

*This behavior can be overridden by setting the auto bus-on (ABO) bit in the CAN\_CTL register. After exiting the bus-off or configuration modes, the CAN error counters are reset.*

## CAN Debug and Test Modes

The CAN module contains test mode features that aid in the debugging of the CAN software and system.

*When these features are used, the CAN module may not be compliant to the CAN specification. All test modes should be enabled or disabled only when the module is in Configuration mode (CCA = 1 in the CAN\_STAT register) or in Suspend mode (CSA = 1 in the CAN\_STAT register).*

The CAN\_DBG.CDE bit is used to gain access to all of the debug features. This bit must be set to enable the test mode, and it must be written first before subsequent writes to the CAN\_DBG register. When the CDE bit is cleared, all debug features are disabled.

When the CAN\_DBG.CDE bit is set, it enables writes to the other bits of the CAN\_DBG register. It also enables these features, which are not compliant with the CAN standard:

- Bit timing registers can be changed anytime, not only during configuration mode. This includes the CAN\_CLK and CAN\_TIMING registers.
- Write access is allowed to the normally read-only CAN\_CEC register.

The CAN\_DBG.MRB bit is used to enable the Readback mode. In this mode, a message transmitted on the CAN bus (or through an internal loopback mode) is received back directly to the internal receive buffer. After a correct transmission, the internal logic treats this as a normal receive message. This feature allows the user to test most of the CAN features without an external device.

The CAN\_DBG.MAA bit allows the CAN module to generate its own acknowledge during the ACK slot of the CAN frame. No external devices or connections are necessary to read back a transmit message. In this mode, the message that is sent is automatically stored in the internal receive buffer. In Auto Acknowledge mode, the module itself transmits the acknowledge. This acknowledge can be programmed to appear on the CAN\_TX pin if CAN\_DBG.DIL = 1 and CAN\_DBG.DTO = 0. If the acknowledge is only going to be used internally, then these test mode bits should be set to CAN\_DBG.DIL = 0 and CAN\_DBG.DTO = 1.

The CAN\_DBG.DIL bit is used to internally enable the transmit output to be routed back to the receive input.

The CAN\_DBG.DTO bit is used to disable the CAN\_TX output pin. When this bit is set, the CAN\_TX pin continuously drives recessive bits.

The CAN\_DBG.DRI bit is used to disable the CAN\_RX input. When set, the internal logic receives recessive bits or receives the internally generated transmit value in the case of the internal loop enabled (CAN\_DBG.DIL = 0). In either case, the value on the CAN\_RX input pin is ignored.

The CAN\_DBG.DEC bit is used to disable the Transmit and Receive Error Counters in the CAN\_CEC register. When this bit is set, the CAN\_CEC holds its current contents and is not allowed to increment or decrement the error counters. This mode does not conform to the CAN specification.

*Writes to the Error Counter registers should be performed in debug mode only. Write access during reception may lead to undefined values. The maximum value which can be written into the Error Counters is 255. Thus, the Error Counter value of 256, which forces the module into the bus off state, cannot be written into the Error Counter registers.*

**Table 21-6:** Common CAN Test Mode Bit Combinations

| MRB | MAA | DIL | DTO | DRI | CDE | Functional Description                                                                                                                                                                                                                |
|-----|-----|-----|-----|-----|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X   | X   | X   | X   | X   | 0   | Normal mode, not debug mode                                                                                                                                                                                                           |
| 0   | X   | X   | X   | X   | X   | No readback of transmit message                                                                                                                                                                                                       |
| 1   | 0   | 1   | 0   | 0   | 1   | Normal transmission on CAN bus line.<br>Read back.<br>External acknowledge from external device required.                                                                                                                             |
| 1   | 1   | 1   | 0   | 0   | 1   | Normal transmission on CAN bus line.<br>Read back.<br>No external acknowledge required.<br>Transmit message and acknowledge are transmitted on CAN bus line.<br>CAN_RX input is enabled.                                              |
| 1   | 1   | 0   | 0   | 0   | 1   | Normal transmission on CAN bus line.<br>Read back.<br>No external acknowledge required.<br>Transmit message and acknowledge are transmitted on CAN bus line.<br>CAN_RX input and internal loop are enabled (internal OR of TX and RX) |



**Table 21-6:** Common CAN Test Mode Bit Combinations (Continued)

| MRB | MAA | DIL | DTO | DRI | CDE | Functional Description                                                                                                                                                                                                |
|-----|-----|-----|-----|-----|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1   | 1   | 0   | 0   | 1   | 1   | Normal transmission on CAN bus line.<br>Read back.<br>No external acknowledge required.<br>Transmit message and acknowledge are transmitted on CAN bus line.<br>CAN_RX input is ignored.<br>Internal loop is enabled. |
| 1   | 1   | 0   | 1   | 1   | 1   | No transmission on CAN bus line.<br>Read back.<br>No external acknowledge required.<br>Neither transmit message nor acknowledge are transmitted on CAN_TX.<br>CAN_RX input is ignored.<br>Internal loop is enabled.   |

## CAN Register Descriptions

Controller Area Network (CAN) contains the following registers.

**Table 21-7:** CAN Module Registers

| Name       | Description                                |
|------------|--------------------------------------------|
| CAN_MC1    | Mailbox Configuration 1 Register           |
| CAN_MD1    | Mailbox Direction 1 Register               |
| CAN_TRS1   | Transmission Request Set 1 Register        |
| CAN_TRR1   | Transmission Request Reset 1 Register      |
| CAN_TA1    | Transmission Acknowledge 1 Register        |
| CAN_AA1    | Abort Acknowledge 1 Register               |
| CAN_RMP1   | Receive Message Pending 1 Register         |
| CAN_RML1   | Receive Message Lost 1 Register            |
| CAN_MBTIF1 | Mailbox Transmit Interrupt Flag 1 Register |

**Table 21-7:** CAN Module Registers (Continued)

| Name        | Description                                              |
|-------------|----------------------------------------------------------|
| CAN_MBRI1F1 | Mailbox Receive Interrupt Flag 1 Register                |
| CAN_MBIM1   | Mailbox Interrupt Mask 1 Register                        |
| CAN_RFH1    | Remote Frame Handling 1 Register                         |
| CAN_OPSS1   | Overwrite Protection/Single Shot Transmission 1 Register |
| CAN_MC2     | Mailbox Configuration 2 Register                         |
| CAN_MD2     | Mailbox Direction 2 Register                             |
| CAN_TRS2    | Transmission Request Set 2 Register                      |
| CAN_TRR2    | Transmission Request Reset 2 Register                    |
| CAN_TA2     | Transmission Acknowledge 2 Register                      |
| CAN_AA2     | Abort Acknowledge 2 Register                             |
| CAN_RMP2    | Receive Message Pending 2 Register                       |
| CAN_RML2    | Receive Message Lost 2 Register                          |
| CAN_MBTIF2  | Mailbox Transmit Interrupt Flag 2 Register               |
| CAN_MBRI1F2 | Mailbox Receive Interrupt Flag 2 Register                |
| CAN_MBIM2   | Mailbox Interrupt Mask 2 Register                        |
| CAN_RFH2    | Remote Frame Handling 2 Register                         |
| CAN_OPSS2   | Overwrite Protection/Single Shot Transmission 2 Register |
| CAN_CLK     | Clock Register                                           |
| CAN_TIMING  | Timing Register                                          |
| CAN_DBG     | Debug Register                                           |
| CAN_STAT    | Status Register                                          |
| CAN_CEC     | Error Counter Register                                   |

**Table 21-7:** CAN Module Registers (Continued)

| Name               | Description                                   |
|--------------------|-----------------------------------------------|
| CAN_GIS            | Global CAN Interrupt Status Register          |
| CAN_GIM            | Global CAN Interrupt Mask Register            |
| CAN_GIF            | Global CAN Interrupt Flag Register            |
| CAN_CTL            | CAN Master Control Register                   |
| CAN_INT            | Interrupt Pending Register                    |
| CAN_MBTD           | Temporary Mailbox Disable Register            |
| CAN_EWR            | Error Counter Warning Level Register          |
| CAN_ESR            | Error Status Register                         |
| CAN_UCCNT          | Universal Counter Register                    |
| CAN_UCRC           | Universal Counter Reload/Capture Register     |
| CAN_UCCNF          | Universal Counter Configuration Mode Register |
| CAN_AMnnL          | Acceptance Mask (L) Register                  |
| CAN_AMnnH          | Acceptance Mask (H) Register                  |
| CAN_MBnn_DATA0     | Mailbox Word 0 Register                       |
| CAN_MBnn_DATA1     | Mailbox Word 1 Register                       |
| CAN_MBnn_DATA2     | Mailbox Word 2 Register                       |
| CAN_MBnn_DATA3     | Mailbox Word 3 Register                       |
| CAN_MBnn_LENGTH    | Mailbox Length Register                       |
| CAN_MBnn_TIMESTAMP | Mailbox Timestamp Register                    |
| CAN_MBnn_ID0       | Mailbox ID 0 Register                         |
| CAN_MBnn_ID1       | Mailbox ID 1 Register                         |

## Mailbox Configuration 1 Register

The CAN\_MC1 register enables mailboxes 0 through 15. Each bit in this register enables or disables the corresponding mailbox. For all bits, set the bit (=1) to enable the mailbox, and clear the bit (=0) to disable the mailbox.

Enabling and disabling mailboxes has an impact on transmit requests. Setting the CAN\_TRS1 bit associated with a disabled mailbox may result in erroneous behavior. Similarly, disabling a mailbox before the associated CAN\_TRS1 bit is reset by the internal logic can cause unpredictable results.

CAN\_MC1: Mailbox Configuration 1 Register - R/W

Reset = 0x0000

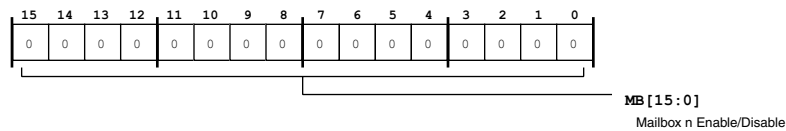


Figure 21-10: CAN\_MC1 Register Diagram

Table 21-8: CAN\_MC1 Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 15:0    | MB       | Mailbox n Enable/Disable. |

## Mailbox Direction 1 Register

The CAN\_MD1 register selects the data transfer direction for mailboxes 0 through 15. Each bit in this register selects receive mode or transmit mode for the corresponding mailbox. For all bits, set the bit (=1) for transmit mode from the mailbox, and clear the bit (=0) for receive mode to the mailbox.

CAN\_MD1: Mailbox Direction 1 Register - R/W

Reset = 0x0000

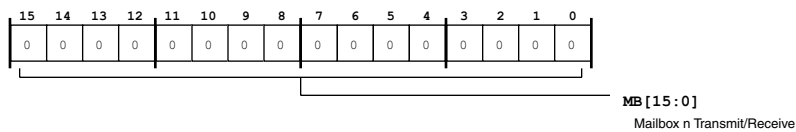


Figure 21-11: CAN\_MD1 Register Diagram

**Table 21-9:** CAN\_MD1 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | MB       | Mailbox n Transmit/Receive. |

### Transmission Request Set 1 Register

The CAN\_TRS1 register requests transmit for mailboxes 0 through 15. Each bit in this register requests transmit for the corresponding mailbox when set (=1). After writing the data and the identifier into the mailbox area, the message is sent after mailbox n is enabled (with the corresponding bit in CAN\_MC1 = 1}, and (subsequently) the corresponding transmit request bit is set (in CAN\_TRS1). When a transmission completes, the corresponding bits in CAN\_TRS1 and in the transmit request reset register (CAN\_TRR1) are cleared.

**CAN\_TRS1: Transmission Request Set 1 Register - R/W**

Reset = 0x0000

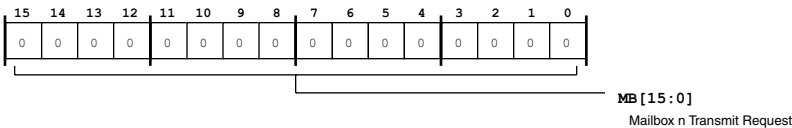


**Figure 21-12:** CAN\_TRS1 Register Diagram

**Table 21-10:** CAN\_TRS1 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | MB       | Mailbox n Transmit Request. |

### Transmission Request Reset 1 Register

The CAN\_TRR1 register requests transmit abort for mailboxes 0 through 15. Each bit in this register requests transmit abort for the corresponding mailbox when set (=1). When a transmission completes, the corresponding bits in the transmit request set register (CAN\_TRS1) and in the CAN\_TRR1 are cleared.

CAN\_TRR1: Transmission Request Reset 1 Register - R/W

Reset = 0x0000

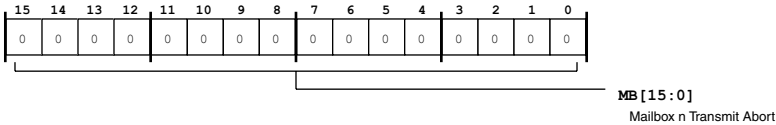


Figure 21-13: CAN\_TRR1 Register Diagram

Table 21-11: CAN\_TRR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 15:0    | MB       | Mailbox n Transmit Abort. |

Transmission Acknowledge 1 Register

The CAN\_TA1 register indicates transmission success for mailboxes 0 through 15. Each bit in this register indicates transmission success for the corresponding mailbox when set (=1).

CAN\_TA1: Transmission Acknowledge 1 Register - R/W

Reset = 0x0000

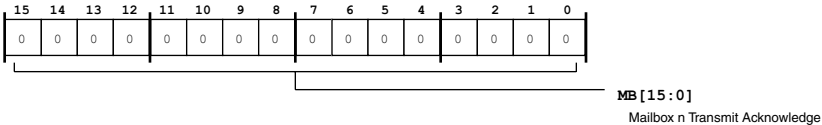


Figure 21-14: CAN\_TA1 Register Diagram

Table 21-12: CAN\_TA1 Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 15:0    | MB       | Mailbox n Transmit Acknowledge. |

Abort Acknowledge 1 Register

The CAN\_AA1 register indicates transmission abort (due to lost arbitration or a CAN error) for mailboxes 0 through 15. Each bit in this register indicates transmission abort for the corresponding mailbox when set (=1).

**CAN\_AA1: Abort Acknowledge 1 Register - R/W**

Reset = 0x0000

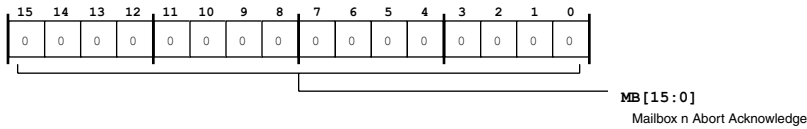


**Figure 21-15:** CAN\_AA1 Register Diagram

**Table 21-13:** CAN\_AA1 Register Fields

| Bit No. | Bit Name | Description/Enumeration      |
|---------|----------|------------------------------|
| 15:0    | MB       | Mailbox n Abort Acknowledge. |

**Receive Message Pending 1 Register**

The CAN\_RMP1 register indicates when a message is pending (unread data) for mailboxes 0 through 15. Each bit in this register indicates the message pending status for the corresponding mailbox when set (=1).

**CAN\_RMP1: Receive Message Pending 1 Register - R/W**

Reset = 0x0000

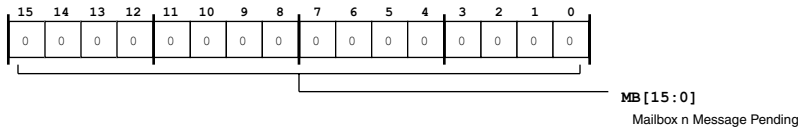


**Figure 21-16:** CAN\_RMP1 Register Diagram

**Table 21-14:** CAN\_RMP1 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 15:0    | MB       | Mailbox n Message Pending. |

**Receive Message Lost 1 Register**

The CAN\_RML1 register indicates when a message is lost---due to a message coming while there is pending data (corresponding CAN\_RMP1 bit set) and overwrite protection is disabled (CAN\_OPSS1 bit cleared)---for mailboxes 0 through 15. Each bit in this register indicates the message lost status for the corresponding mailbox when set (=1).

CAN\_RML1: Receive Message Lost 1 Register - R/W

Reset = 0x0000

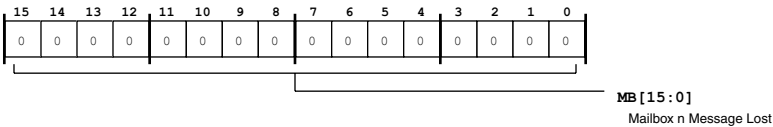


Figure 21-17: CAN\_RML1 Register Diagram

Table 21-15: CAN\_RML1 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | MB       | Mailbox n Message Lost. |

Mailbox Transmit Interrupt Flag 1 Register

The CAN\_MBTIF1 register indicates when a transmit interrupt is pending---due to successful transmission (corresponding CAN\_TA1 bit set) and the interrupt is enabled (corresponding CAN\_MBIM1 bit set)---for mailboxes 0 through 15. Each bit in this register indicates the transmit interrupt pending status for the corresponding mailbox when set (=1). When any bit in CAN\_MBTIF1 is set, the CAN transmit interrupt request is raised (CAN\_INT.MBTIRQ bit set). To clear the interrupt request, all of the set bits in CAN\_MBTIF1 must be cleared by software (W1C). Also, software must clear the associated bits set in CAN\_TA1 or set the associated bits in CAN\_TRS1 bit to clear the interrupt source asserting the bits in CAN\_MBTIF1.

CAN\_MBTIF1: Mailbox Transmit Interrupt Flag 1 Register - R/W

Reset = 0x0000

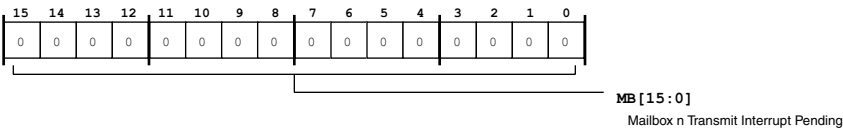


Figure 21-18: CAN\_MBTIF1 Register Diagram

Table 21-16: CAN\_MBTIF1 Register Fields

| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 15:0    | MB       | Mailbox n Transmit Interrupt Pending. |



## Mailbox Receive Interrupt Flag 1 Register

The CAN\_MBIF1 register indicates when a receive interrupt is pending---due to successful reception (corresponding CAN\_RMP1 bit set) and the interrupt is enabled (corresponding CAN\_MBIM1 bit set)---for mailboxes 0 through 15. Each bit in this register indicates the receive interrupt pending status for the corresponding mailbox when set (=1). When any bit in CAN\_MBIF1 is set, the CAN receive interrupt request is raised (CAN\_INT.MBIRQ bit set). To clear the interrupt request, all of the set bits in CAN\_RMP1 must be cleared by software, then the associated bits set in CAN\_MBIF1 must be cleared (W1C).

CAN\_MBIF1: Mailbox Receive Interrupt Flag 1 Register - R/W

Reset = 0x0000

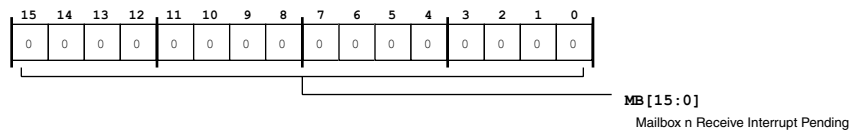


Figure 21-19: CAN\_MBIF1 Register Diagram

Table 21-17: CAN\_MBIF1 Register Fields

| Bit No. | Bit Name | Description/Enumeration              |
|---------|----------|--------------------------------------|
| 15:0    | MB       | Mailbox n Receive Interrupt Pending. |

## Mailbox Interrupt Mask 1 Register

The CAN\_MBIM1 register enables transmit and receive interrupts for mailboxes 0 through 15. Each bit in this register requests enables the transmit or receive interrupt for the corresponding mailbox when set (=1).

CAN\_MBIM1: Mailbox Interrupt Mask 1 Register - R/W

Reset = 0x0000

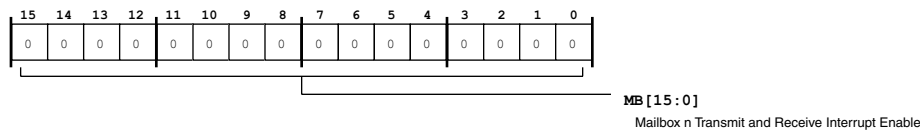


Figure 21-20: CAN\_MBIM1 Register Diagram

Table 21-18: CAN\_MBIM1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                          |
|---------|----------|--------------------------------------------------|
| 15:0    | MB       | Mailbox n Transmit and Receive Interrupt Enable. |

Remote Frame Handling 1 Register

The CAN\_RFH1 register enables remote frame handling for mailboxes 0 through 15. Each bit in this register enables remote frame handling for the corresponding mailbox when set (=1). Note that enabling this bit affects transmit and receive operations for mailboxes. For more information about remote frame handling, see the CAN Operating Modes sections, describing transmit and receive operations.

CAN\_RFH1: Remote Frame Handling 1 Register - R/W

Reset = 0x0000

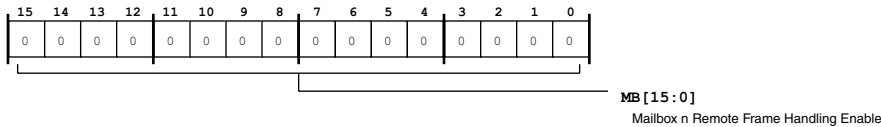


Figure 21-21: CAN\_RFH1 Register Diagram

Table 21-19: CAN\_RFH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                 |
|---------|----------|-----------------------------------------|
| 15:0    | MB       | Mailbox n Remote Frame Handling Enable. |

Overwrite Protection/Single Shot Transmission 1 Register

The CAN\_OPSS1 register enables overwrite protection for mailboxes 0 through 15. Each bit in this register enables overwrite protection for the corresponding mailbox when set (=1). Note that enabling this bit affects transmit and receive operations for mailboxes. For more information about remote overwrite protection, see the detailed feature description in the CAN Functional Description section. For more information about how this feature affects transmit and receive operations, see the CAN Operating Modes sections, describing transmit and receive operations.

**CAN\_OPSS1: Overwrite Protection/Single Shot Transmission 1 Register - R/W**

Reset = 0x0000

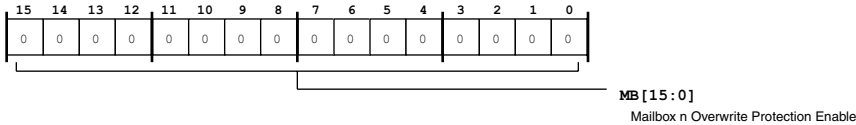


**Figure 21-22:** CAN\_OPSS1 Register Diagram

**Table 21-20:** CAN\_OPSS1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 15:0    | MB       | Mailbox n Overwrite Protection Enable. |

**Mailbox Configuration 2 Register**

The CAN\_MC2 register enables mailboxes 16 through 31. Each bit in this register enables or disables the corresponding mailbox. For all bits, set the bit (=1) to enable the mailbox, and clear the bit (=0) to disable the mailbox.

Enabling and disabling mailboxes has an impact on transmit requests. Setting the CAN\_TRS2 bit associated with a disabled mailbox may result in erroneous behavior. Similarly, disabling a mailbox before the associated CAN\_TRS2 bit is reset by the internal logic can cause unpredictable results.

**CAN\_MC2: Mailbox Configuration 2 Register - R/W**

Reset = 0x0000

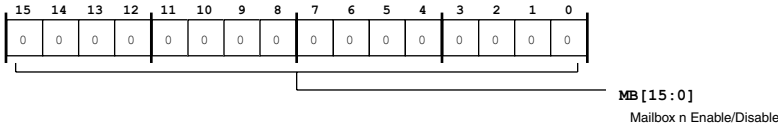


**Figure 21-23:** CAN\_MC2 Register Diagram

**Table 21-21:** CAN\_MC2 Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 15:0    | MB       | Mailbox n Enable/Disable. |

## Mailbox Direction 2 Register

The CAN\_MD2 register selects the data transfer direction for mailboxes 16 through 32. Each bit in this register selects receive mode or transmit mode for the corresponding mailbox. For all bits, set the bit (=1) for transmit mode from the mailbox, and clear the bit (=0) for receive mode to the mailbox.

CAN\_MD2: Mailbox Direction 2 Register - R/W

Reset = 0x0000

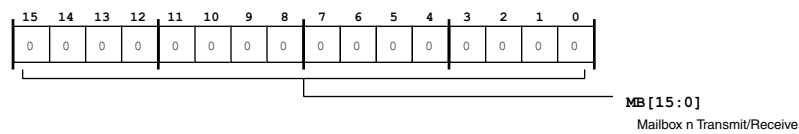


Figure 21-24: CAN\_MD2 Register Diagram

Table 21-22: CAN\_MD2 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | MB       | Mailbox n Transmit/Receive. |

## Transmission Request Set 2 Register

The CAN\_TRS2 register requests transmit for mailboxes 16 through 32. Each bit in this register requests transmit for the corresponding mailbox when set (=1). After writing the data and the identifier into the mailbox area, the message is sent after mailbox n is enabled (with the corresponding bit in CAN\_MC2 = 1}, and (subsequently) the corresponding transmit request bit is set (in CAN\_TRS2). When a transmission completes, the corresponding bits in CAN\_TRS2 and in the transmit request reset register (CAN\_TRR2) are cleared.

CAN\_TRS2: Transmission Request Set 2 Register - R/W

Reset = 0x0000

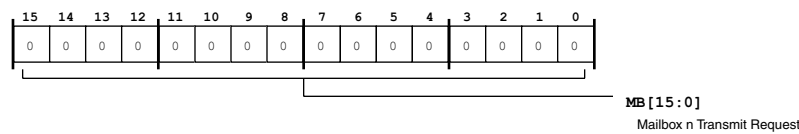


Figure 21-25: CAN\_TRS2 Register Diagram

Table 21-23: CAN\_TRS2 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | MB       | Mailbox n Transmit Request. |

## Transmission Request Reset 2 Register

The CAN\_TRR2 register requests transmit abort for mailboxes 16 through 32. Each bit in this register requests transmit abort for the corresponding mailbox when set (=1). When a transmission completes, the corresponding bits in the transmit request set register (CAN\_TRS2) and in the CAN\_TRR2 are cleared.

CAN\_TRR2: Transmission Request Reset 2 Register - R/W

Reset = 0x0000

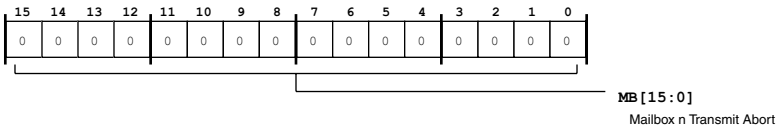


Figure 21-26: CAN\_TRR2 Register Diagram

Table 21-24: CAN\_TRR2 Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 15:0    | MB       | Mailbox n Transmit Abort. |

## Transmission Acknowledge 2 Register

The CAN\_TA2 register indicates transmission success for mailboxes 16 through 32. Each bit in this register indicates transmission success for the corresponding mailbox when set (=1).

CAN\_TA2: Transmission Acknowledge 2 Register - R/W

Reset = 0x0000

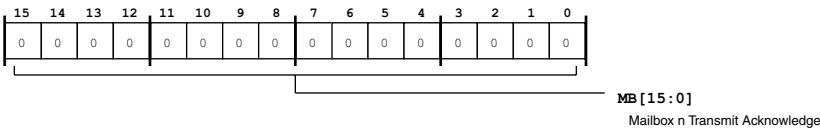


Figure 21-27: CAN\_TA2 Register Diagram

Table 21-25: CAN\_TA2 Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 15:0    | MB       | Mailbox n Transmit Acknowledge. |

## Abort Acknowledge 2 Register

The CAN\_AA2 register indicates transmission abort (due to lost arbitration or a CAN error) for mailboxes 16 through 32. Each bit in this register indicates transmission abort for the corresponding mailbox when set (=1).

CAN\_AA2: Abort Acknowledge 2 Register - R/W

Reset = 0x0000

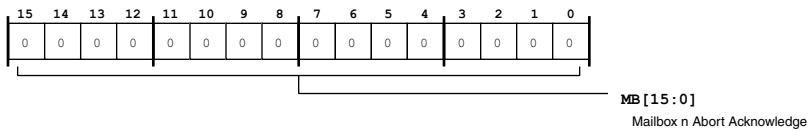


Figure 21-28: CAN\_AA2 Register Diagram

Table 21-26: CAN\_AA2 Register Fields

| Bit No. | Bit Name | Description/Enumeration      |
|---------|----------|------------------------------|
| 15:0    | MB       | Mailbox n Abort Acknowledge. |

## Receive Message Pending 2 Register

The CAN\_RMP2 register indicates when a message is pending (unread data) for mailboxes 16 through 32. Each bit in this register indicates the message pending status for the corresponding mailbox when set (=1).

CAN\_RMP2: Receive Message Pending 2 Register - R/W

Reset = 0x0000

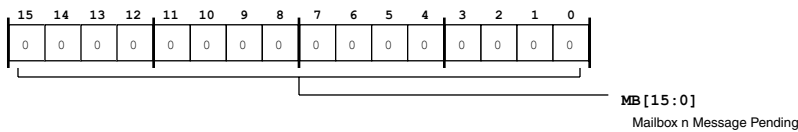


Figure 21-29: CAN\_RMP2 Register Diagram

Table 21-27: CAN\_RMP2 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 15:0    | MB       | Mailbox n Message Pending. |

## Receive Message Lost 2 Register

The CAN\_RML2 register indicates when a message is lost---due to a message coming while there is pending data (corresponding CAN\_RMP2 bit set) and overwrite protection is disabled (CAN\_OPSS2 bit cleared)---for mailboxes 16 through 32. Each bit in this register indicates the message lost status for the corresponding mailbox when set (=1).

**CAN\_RML2: Receive Message Lost 2 Register - R/W**

Reset = 0x0000

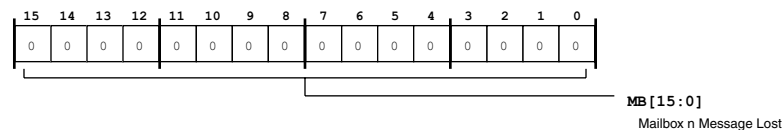


**Figure 21-30:** CAN\_RML2 Register Diagram

**Table 21-28:** CAN\_RML2 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | MB       | Mailbox n Message Lost. |

## Mailbox Transmit Interrupt Flag 2 Register

The CAN\_MBTIF2 register indicates when a transmit interrupt is pending---due to successful transmission (corresponding CAN\_TA2 bit set) and the interrupt is enabled (corresponding CAN\_MBIM2 bit set)---for mailboxes 16 through 32. Each bit in this register indicates the transmit interrupt pending status for the corresponding mailbox when set (=1). When any bit in CAN\_MBTIF2 is set, the CAN transmit interrupt request is raised (CAN\_INT.MBTIRQ bit set). To clear the interrupt request, all of the set bits in CAN\_MBTIF2 must be cleared by software (W1C). Also, software must clear the associated bits set in CAN\_TA2 or set the associated bits in CAN\_TRS2 bit to clear the interrupt source asserting the bits in CAN\_MBTIF2.

**CAN\_MBTIF2: Mailbox Transmit Interrupt Flag 2 Register - R/W**

Reset = 0x0000

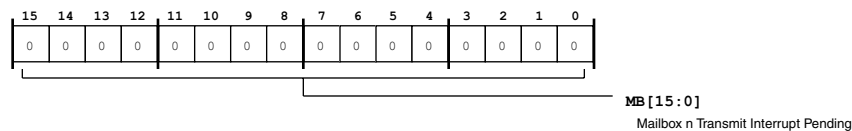


**Figure 21-31:** CAN\_MBTIF2 Register Diagram

Table 21-29: CAN\_MBTIF2 Register Fields

| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 15:0    | MB       | Mailbox n Transmit Interrupt Pending. |

Mailbox Receive Interrupt Flag 2 Register

The CAN\_MBIF2 register indicates when a receive interrupt is pending---due to successful reception (corresponding CAN\_RMP2 bit set) and the interrupt is enabled (corresponding CAN\_MBIM2 bit set)---for mailboxes 16 through 32. Each bit in this register indicates the receive interrupt pending status for the corresponding mailbox when set (=1). When any bit in CAN\_MBIF2 is set, the CAN receive interrupt request is raised (CAN\_INT.MBIRQ bit set). To clear the interrupt request, all of the set bits in CAN\_RMP2 must be cleared by software, then the associated bits set in CAN\_MBIF2 must be cleared (W1C).

CAN\_MBIF2: Mailbox Receive Interrupt Flag 2 Register - R/W

Reset = 0x0000

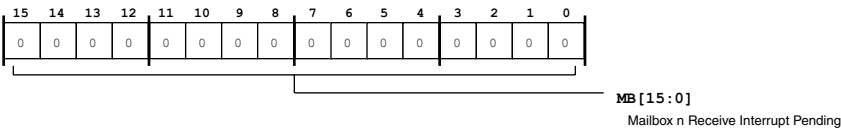


Figure 21-32: CAN\_MBIF2 Register Diagram

Table 21-30: CAN\_MBIF2 Register Fields

| Bit No. | Bit Name | Description/Enumeration              |
|---------|----------|--------------------------------------|
| 15:0    | MB       | Mailbox n Receive Interrupt Pending. |

Mailbox Interrupt Mask 2 Register

The CAN\_MBIM2 register enables transmit and receive interrupts for mailboxes 16 through 32. Each bit in this register requests enables the transmit or receive interrupt for the corresponding mailbox when set (=1).



**CAN\_MBIM2: Mailbox Interrupt Mask 2 Register - R/W**

Reset = 0x0000

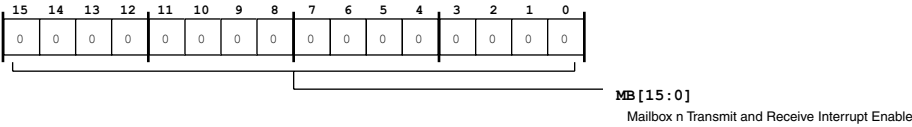


**Figure 21-33:** CAN\_MBIM2 Register Diagram

**Table 21-31:** CAN\_MBIM2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                          |
|---------|----------|--------------------------------------------------|
| 15:0    | MB       | Mailbox n Transmit and Receive Interrupt Enable. |

**Remote Frame Handling 2 Register**

The CAN\_RFH2 register enables remote frame handling for mailboxes 16 through 32. Each bit in this register enables remote frame handling for the corresponding mailbox when set (=1). Note that enabling this bit affects transmit and receive operations for mailboxes. For more information about remote frame handling, see the CAN Operating Modes sections, describing transmit and receive operations.

**CAN\_RFH2: Remote Frame Handling 2 Register - R/W**

Reset = 0x0000

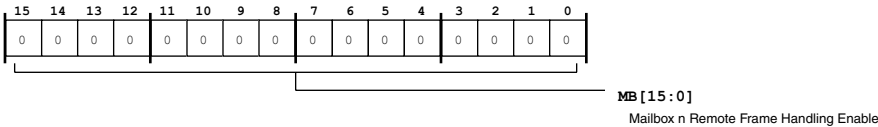


**Figure 21-34:** CAN\_RFH2 Register Diagram

**Table 21-32:** CAN\_RFH2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                 |
|---------|----------|-----------------------------------------|
| 15:0    | MB       | Mailbox n Remote Frame Handling Enable. |

**Overwrite Protection/Single Shot Transmission 2 Register**

The CAN\_OPSS2 register enables overwrite protection for mailboxes 16 through 32. Each bit in this register enables overwrite protection for the corresponding mailbox when set (=1). Note that enabling this bit affects transmit and receive operations for mailboxes. For more information about remote overwrite

protection, see the detailed feature description in the CAN Functional Description section. For more information about how this feature affects transmit and receive operations, see the CAN Operating Modes sections, describing transmit and receive operations.

CAN\_OPSS2: Overwrite Protection/Single Shot Transmission 2 Register - R/W

Reset = 0x0000

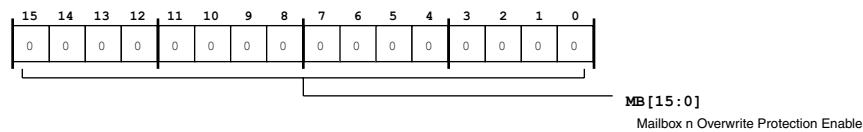


Figure 21-35: CAN\_OPSS2 Register Diagram

Table 21-33: CAN\_OPSS2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 15:0    | MB       | Mailbox n Overwrite Protection Enable. |

Clock Register

The CAN\_CLK register select the bit rate prescaler for calculating the time quantum (TQ), which is used to derive the CAN clock from the system clock (SCLK). For more information about bit timing and clock operation, see the CAN Operating Modes section.

CAN\_CLK: Clock Register - R/W

Reset = 0x0000

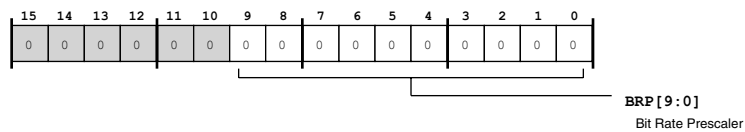


Figure 21-36: CAN\_CLK Register Diagram

**Table 21-34:** CAN\_CLK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:0     | BRP      | <p>Bit Rate Prescaler.</p> <p>The CAN_CLK.BRP bits select the bit rate prescaler value, which is used to calculate the time quantum for CAN bit timing. The formula using CAN_CLK.BRP to calculate the time quantum is:</p> $TQ = (BRP + 1) / SCLK$ <p>Note that it is recommended that the CAN_CLK.BRP value be greater than or equal to 4. For more information about bit timing, see the Operating Modes section.</p> |

## Timing Register

The CAN\_TIMING register select the time segments, sampling, and synchronization for CAN bit timing. For more information about bit timing and clock operation, see the CAN Operating Modes section.

**CAN\_TIMING: Timing Register - R/W**

Reset = 0x0000

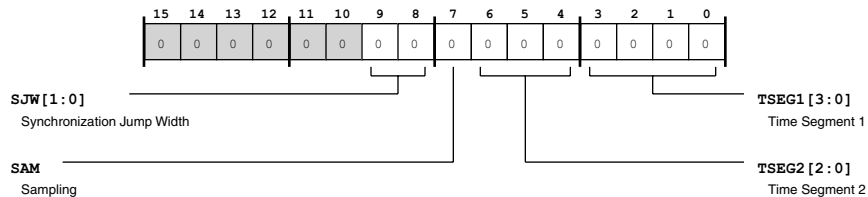


**Figure 21-37:** CAN\_TIMING Register Diagram

**Table 21-35:** CAN\_TIMING Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | SJW      | <p>Synchronization Jump Width.</p> <p>The CAN_TIMING.SJW bits select the maximum number of time quanta, ranging from 1 to 4(SJW + 1). This selection allows for a re-synchronization attempt when the CAN detects a recessive-to-dominant edge outside the synchronization segment. The re-synchronization automatically moves the sampling point such that the CAN bit is still handled properly. Note that the CAN_TIMING.SJW value should not exceed CAN_TIMING.TSEG2 or CAN_TIMING.TSEG1.</p> |

**Table 21-35:** CAN\_TIMING Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | SAM      | <p>Sampling.</p> <p>The CAN_TIMING.SAM bit selects whether the CAN performs normal sampling (once at the sampling point described by the CAN_TIMING register) or performs over sampling. If CAN_TIMING.SAM is set, the CAN over samples the input signal at three times at the SCLK rate. The resulting value is generated by a majority decision of the three sample values. Note that the CAN_TIMING.SAM bit should always be cleared if the CAN_CLK.BRP value is less than 4.</p> |
| 6:4     | TSEG2    | <p>Time Segment 2.</p> <p>The CAN_TIMING.TSEG2 bits and CAN_TIMING.TSEG1 bits control how many time quanta of which the CAN bits consist, resulting in the CAN bit rate. For more information about bit timing and clock operation, see the CAN Operating Modes section. Note that the CAN_TIMING.TSEG1 value should always be greater than or equal to the CAN_TIMING.TSEG2 value.</p>                                                                                              |
| 3:0     | TSEG1    | <p>Time Segment 1.</p> <p>The CAN_TIMING.TSEG1 bits and CAN_TIMING.TSEG2 bits control how many time quanta of which the CAN bits consist, resulting in the CAN bit rate. For more information about bit timing and clock operation, see the CAN Operating Modes section. Note that the CAN_TIMING.TSEG1 value should always be greater than or equal to the CAN_TIMING.TSEG2 value.</p>                                                                                              |

## Debug Register

The CAN\_DBG register controls CAN debug modes, including CAN\_TX and CAN\_RX pin enable/disable.

**CAN\_DBG: Debug Register - R/W**

Reset = 0x0008

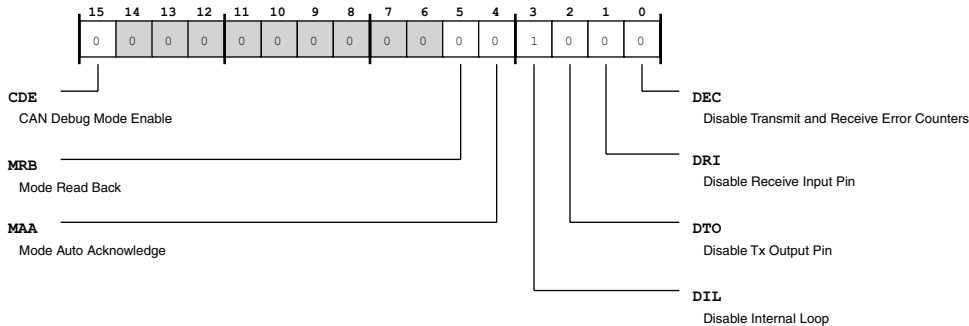


**Figure 21-38:** CAN\_DBG Register Diagram

**Table 21-36:** CAN\_DBG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | CDE      | CAN Debug Mode Enable.<br>The CAN_DBG.CDE bit enables debug mode. This bit must be written first before subsequent writes to the CAN_DBG register. When the CAN_DBG.CDE bit is cleared, all CAN debug features are disabled. |
|         |          | 0      Disable Debug Mode                                                                                                                                                                                                    |
|         |          | 1      Enable Debug Mode                                                                                                                                                                                                     |
| 5       | MRB      | Mode Read Back.<br>The CAN_DBG.MRB bit enables read back mode. When enabled, a message transmitted on the CAN bus or through an internal loop back mode is received back directly to the internal receive buffer.            |
|         |          | 0      Disable Read Back Mode                                                                                                                                                                                                |
|         |          | 1      Enable Read Back Mode                                                                                                                                                                                                 |

**Table 21-36:** CAN\_DBG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | MAA      | Mode Auto Acknowledge.<br>The CAN_DBG.MAA bit enables mode auto acknowledge, allowing the CAN to generate its own acknowledge during the ACK slot of the CAN frame. The CAN_DBG.MAA acknowledge appears on the CAN_TX pin if CAN_DBG.DIL =1 and CAN_DBG.DTO =0. If the acknowledge is only going to be used internally, these test mode bits should be set to CAN_DBG.DIL = 0 and CAN_DBG.DTO =1. |
|         |          | 0      Disable Auto Acknowledge Mode                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1      Enable Auto Acknowledge Mode                                                                                                                                                                                                                                                                                                                                                               |
| 3       | DIL      | Disable Internal Loop.<br>The CAN_DBG.DIL bit disables internal loop mode, which routes the transmit output to the receive input.                                                                                                                                                                                                                                                                 |
|         |          | 0      Enable Internal Loop                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Disable Internal Loop                                                                                                                                                                                                                                                                                                                                                                      |
| 2       | DTO      | Disable Tx Output Pin.<br>The CAN_DBG.DTO bit disables the CAN_TX pin.                                                                                                                                                                                                                                                                                                                            |
|         |          | 0      Enable Tx Output Pin                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Disable Tx Output Pin, Drive Recessive                                                                                                                                                                                                                                                                                                                                                     |
| 1       | DRI      | Disable Receive Input Pin.<br>The CAN_DBG.DRI bit disables the CAN_RX pin.                                                                                                                                                                                                                                                                                                                        |
|         |          | 0      Enable Rx Input Pin                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Disable Rx Input Pin, Drive Recessive Internally                                                                                                                                                                                                                                                                                                                                           |

**Table 21-36:** CAN\_DBG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | DEC      | Disable Transmit and Receive Error Counters.<br>The CAN_DBG.DEC bit disables the transmit and receive error counters in the CAN_CEC register. When set, the CAN_CEC holds its current contents and is not allowed to increment or decrement the error counters. Note that this mode does not conform to the CAN specification. |
|         |          | 0 Enable CEC Tx and Rx Error Counters                                                                                                                                                                                                                                                                                          |
|         |          | 1 Disable CEC Tx and Rx Error Counters                                                                                                                                                                                                                                                                                         |

### Status Register

The CAN\_STAT register indicates status for CAN modes and error conditions.

**CAN\_STAT: Status Register - R/W**

Reset = 0x0000

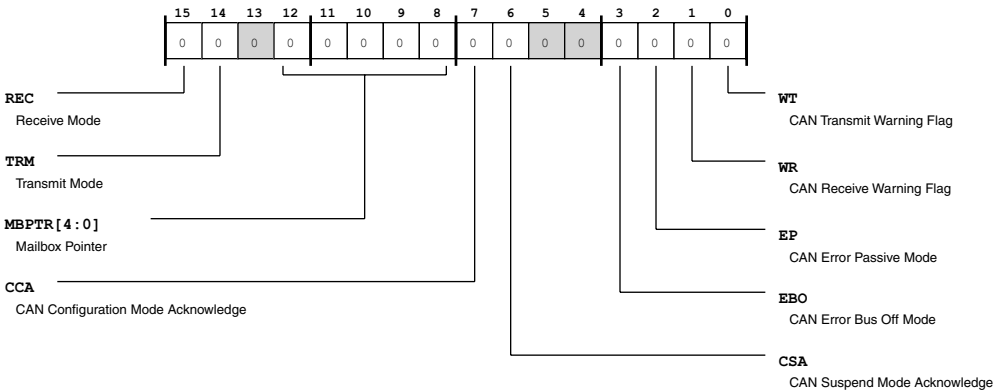


**Figure 21-39:** CAN\_STAT Register Diagram

**Table 21-37:** CAN\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | REC      | Receive Mode.<br>The CAN_STAT.REC bit indicates whether the CAN is in receive mode.                                                                                     |
|         |          | 0 Not in Receive Mode                                                                                                                                                   |
|         |          | 1 Receive Mode                                                                                                                                                          |
| 14      | TRM      | Transmit Mode.<br>The CAN_STAT.TRM bit indicates whether the CAN is in transmit mode.                                                                                   |
|         |          | 0 Not in Transmit Mode                                                                                                                                                  |
|         |          | 1 Transmit Mode                                                                                                                                                         |
| 12:8    | MBPTR    | Mailbox Pointer.<br>The CAN_STAT.MBPTR bits represent the mailbox number of the current transmit message. After a successful transmission, these bits remain unchanged. |
|         |          | 0 Processing Mailbox 0 Message                                                                                                                                          |
|         |          | ... ...                                                                                                                                                                 |
|         |          | 31 Processing Mailbox 31 Message                                                                                                                                        |
| 7       | CCA      | CAN Configuration Mode Acknowledge.<br>The CAN_STAT.CCA bit indicates whether the CAN is in configuration mode.                                                         |
|         |          | 0 Not in Configuration Mode                                                                                                                                             |
|         |          | 1 Configuration mode                                                                                                                                                    |
| 6       | CSA      | CAN Suspend Mode Acknowledge.<br>The CAN_STAT.CSA bit indicates whether the CAN is in suspend mode.                                                                     |
|         |          | 0 Not in Suspend Mode                                                                                                                                                   |
|         |          | 1 Suspend mode                                                                                                                                                          |



**Table 21-37:** CAN\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------|
| 3       | EBO      | CAN Error Bus Off Mode.<br>The CAN_STAT.EBO bit indicates whether the CAN is in error bus off mode.                       |
|         |          | 0 TXECNT Below 256                                                                                                        |
|         |          | 1 TXECNT Above Bus Off Limit                                                                                              |
| 2       | EP       | CAN Error Passive Mode.<br>The CAN_STAT.EP bit indicates whether the CAN is in error passive mode.                        |
|         |          | 0 TXECNT and RXECNT Below 128                                                                                             |
|         |          | 1 TXECNT or RXECNT Above EP Level                                                                                         |
| 1       | WR       | CAN Receive Warning Flag.<br>The CAN_STAT.WR bit indicates whether the CAN has detected a receive warning flag condition. |
|         |          | 0 RXECNT Below Limit                                                                                                      |
|         |          | 1 RXECNT at Limit                                                                                                         |
| 0       | WT       | CAN Transmit Warning Flag.<br>The CAN_STAT.WT bit indicates whether the CAN detected a transmit warning flag condition.   |
|         |          | 0 TXECNT Below Limit                                                                                                      |
|         |          | 1 TXECNT at Limit                                                                                                         |

## Error Counter Register

The CAN\_CEC register, CAN\_ESR register, and CAN\_EWR register control CAN warnings and errors. For detailed information about error and warning operations, see the Event Control section.

The CAN\_CEC register holds an error counter for transmit (CAN\_CEC.TXECNT) and an error counter for receive (CAN\_CEC.RXECNT). After initialization, both counters are 0. Each time a bus error occurs, one of the counters is incremented by either 1 or 8, depending on the error situation (documented in Version 2.0 of CAN Specification). Successful transmit and receive operations decrement the respective counter by 1.

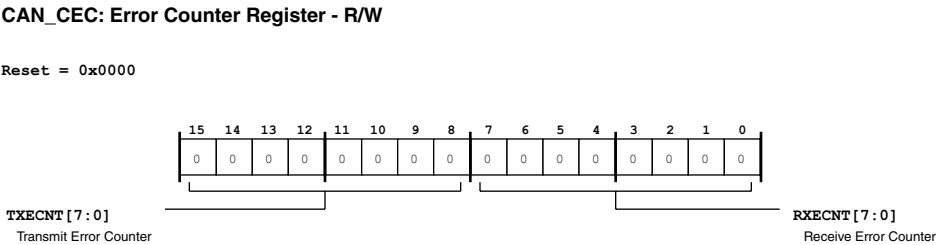


Figure 21-40: CAN\_CEC Register Diagram

Table 21-38: CAN\_CEC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | TXECNT   | Transmit Error Counter.<br>The CAN_CEC.TXECNT bits hold the transmit error counter, which is incremented for errors (by either 1 or 8) and is decremented (by 1) for successful transmit operations. |
| 7:0     | RXECNT   | Receive Error Counter.<br>The CAN_CEC.RXECNT bits hold the receive error counter, which is incremented for errors (by either 1 or 8) and is decremented (by 1) for successful receive operations.    |

Global CAN Interrupt Status Register

The CAN\_GIS register, CAN\_GIF register, and CAN\_GIM register control CAN interrupts. For detailed information about interrupt operations, see the Event Control section.

The CAN\_GIS register holds the interrupt status. All bits in this register are W1C.

**CAN\_GIS: Global CAN Interrupt Status Register - R/W**

Reset = 0x0000

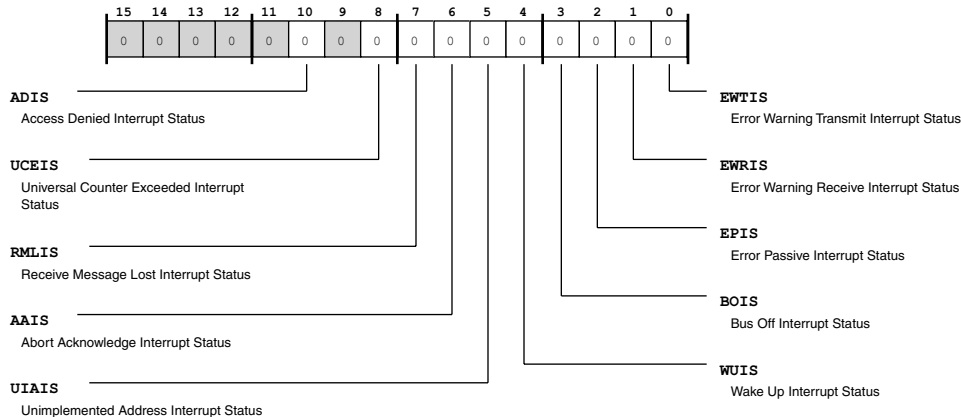


**Figure 21-41:** CAN\_GIS Register Diagram

**Table 21-39:** CAN\_GIS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | ADIS     | Access Denied Interrupt Status.<br>The CAN_GIS.ADIS bit indicates when at least one access to the mailbox RAM occurred during a data update by internal logic.                                                                                      |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                              |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                 |
| 8       | UCEIS    | Universal Counter Exceeded Interrupt Status.<br>The CAN_GIS.UCEIS bit indicates when there has been an overflow of the universal counter (in time stamp mode or event counter mode) or the counter has reached the value 0x0000 (in watchdog mode). |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                              |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                 |

**Table 21-39:** CAN\_GIS Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | RMLIS    | Receive Message Lost Interrupt Status.<br>The CAN_GIS.RMLIS bit indicates when a message is received for a mailbox that currently contains unread data. At least one bit in the receive message lost register (CAN_RML1 or CAN_RML2) is set. If the bit in CAN_GIS (and CAN_GIF) is reset and there is at least one bit in CAN_RML1 or CAN_RML2 still set, the bit in CAN_GIF (and CAN_GIF) is not set again. The internal interrupt source signal is only active if a new bit in CAN_RML1 or CAN_RML2 is set.            |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 6       | AAIS     | Abort Acknowledge Interrupt Status.<br>The CAN_GIS.AAIS bit indicates when At least one abort acknowledge bit is set in the CAN_AA1 or the CAN_AA2 registers. If the bit in CAN_GIS (and CAN_GIF) is reset and there is at least one bit in CAN_AA1 or CAN_AA2 still set, the bit in CAN_GIS (and CAN_GIF) is not set again. The internal interrupt source signal is only active if a new bit in CAN_AA1 or CAN_AA2 is set. The abort acknowledge bits maintain state even after the corresponding mailbox n is disabled. |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 5       | UIAIS    | Unimplemented Address Interrupt Status.<br>The CAN_GIS.UIAIS bit indicates when there was a processor core access to an address that is not implemented in the CAN.                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 4       | WUIS     | Wake Up Interrupt Status.<br>The CAN_GIS.WUIS bit indicates when the CAN has left the sleep mode because of detected activity on the CAN bus line.                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 0 No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Table 21-39:** CAN\_GIS Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | BOIS     | Bus Off Interrupt Status.<br>The CAN_GIS.BOIS bit indicates when the CAN has entered the bus-off state. This interrupt source is active if the status of the CAN changes from normal operation mode to the bus-off mode. If the bit in CAN_GIS (and CAN_GIF) is reset and the bus-off mode is still active, this bit is not set again. If the module leaves the bus-off mode, the bit inCAN_GIS (and CAN_GIF) remains set.                             |
|         |          | 0No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 2       | EPIS     | Error Passive Interrupt Status.<br>The CAN_GIS.EPIS bit indicates when the CAN has entered the error passive state. This interrupt source is active if the status of the CAN changes from the error active mode to the error passive mode. If the bit in CAN_GIS (and CAN_GIF) is reset and the error passive mode is still active, this bit is not set again. If the CAN leaves the error passive mode, the bit in CAN_GIS (and CAN_GIF) remains set. |
|         |          | 0No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 1       | EWRIS    | Error Warning Receive Interrupt Status.<br>The CAN_GIS.EWRIS bit indicates when the CAN_CEC.RXECNT has reached the warning limit. If the bit in CAN_GIS (and CAN_GIF) is reset and the error warning mode is still active, this bit is not set again. If the CAN leaves the error warning mode, the bit in CAN_GIS (and CAN_GIF) remains set.                                                                                                          |
|         |          | 0No Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1Interrupt Pending                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**Table 21-39:** CAN\_GIS Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | EWTIS    | Error Warning Transmit Interrupt Status.<br>The CAN_GIS.EWTIS bit indicates when the CAN_CEC.TXECNT has reached the warning limit. If the bit in CAN_GIS (and CAN_GIF) is reset and the error warning mode is still active, this bit is not set again. If the CAN leaves the error warning mode, the bit in CAN_GIS (and CAN_GIF) remains set. |
|         |          | 0                      No Interrupt Pending                                                                                                                                                                                                                                                                                                    |
|         |          | 1                      Interrupt Pending                                                                                                                                                                                                                                                                                                       |

**Global CAN Interrupt Mask Register**

The CAN\_GIM register, CAN\_GIF register, and CAN\_GIF register control CAN interrupts. For detailed information about interrupt operations, see the Event Control section.

The CAN\_GIM register holds the interrupt mask. The interrupt mask bits only affect the content of the CAN\_GIF register. If the mask bit is not set (enabled/unmasked), the corresponding flag bit is not set when the event occurs.

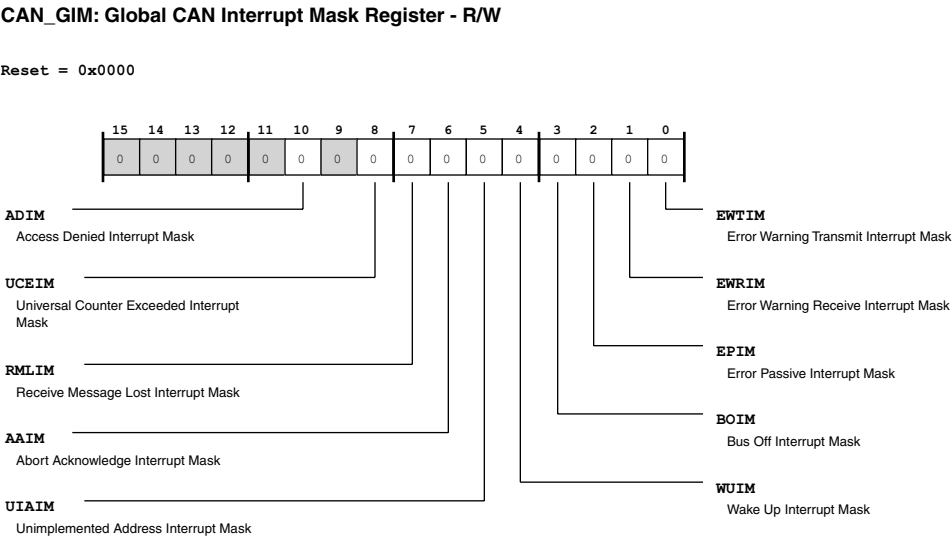


**Figure 21-42:** CAN\_GIM Register Diagram

**Table 21-40:** CAN\_GIM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| 10      | ADIM     | Access Denied Interrupt Mask.<br>The CAN_GIM.ADIM bit enables (unmasks) the access denied interrupt.                            |
|         |          | 0      Disable Interrupt (Mask)                                                                                                 |
|         |          | 1      Enable Interrupt (Unmask)                                                                                                |
| 8       | UCEIM    | Universal Counter Exceeded Interrupt Mask.<br>The CAN_GIM.UCEIM bit enables (unmasks) the universal counter exceeded interrupt. |
|         |          | 0      Disable Interrupt (Mask)                                                                                                 |
|         |          | 1      Enable Interrupt (Unmask)                                                                                                |
| 7       | RMLIM    | Receive Message Lost Interrupt Mask.<br>The CAN_GIM.RMLIM bit enables (unmasks) the receive message lost interrupt.             |
|         |          | 0      Disable Interrupt (Mask)                                                                                                 |
|         |          | 1      Enable Interrupt (Unmask)                                                                                                |
| 6       | AAIM     | Abort Acknowledge Interrupt Mask.<br>The CAN_GIM.AAIM bit enables (unmasks) the abort acknowledge interrupt.                    |
|         |          | 0      Disable Interrupt (Mask)                                                                                                 |
|         |          | 1      Enable Interrupt (Unmask)                                                                                                |
| 5       | UIAIM    | Unimplemented Address Interrupt Mask.<br>The CAN_GIM.UIAIM bit enables (unmasks) the unimplemented address interrupt.           |
|         |          | 0      Disable Interrupt (Mask)                                                                                                 |
|         |          | 1      Enable Interrupt (Unmask)                                                                                                |

**Table 21-40:** CAN\_GIM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 4       | WUIM     | Wake Up Interrupt Mask.<br>The CAN_GIM.WUIM bit enables (unmasks) the wake up interrupt.                                |
|         |          | 0 Disable Interrupt (Mask)                                                                                              |
|         |          | 1 Enable Interrupt (Unmask)                                                                                             |
| 3       | BOIM     | Bus Off Interrupt Mask.<br>The CAN_GIM.BOIM bit enables (unmasks) the bus off interrupt.                                |
|         |          | 0 Disable Interrupt (Mask)                                                                                              |
|         |          | 1 Enable Interrupt (Unmask)                                                                                             |
| 2       | EPIM     | Error Passive Interrupt Mask.<br>The CAN_GIM.EPIM bit enables (unmasks) the error passive mode interrupt.               |
|         |          | 0 Disable Interrupt (Mask)                                                                                              |
|         |          | 1 Enable Interrupt (Unmask)                                                                                             |
| 1       | EWRIM    | Error Warning Receive Interrupt Mask.<br>The CAN_GIM.EWRIM bit enables (unmasks) the error warning receive interrupt.   |
|         |          | 0 Disable Interrupt (Mask)                                                                                              |
|         |          | 1 Enable Interrupt (Unmask)                                                                                             |
| 0       | EWTIM    | Error Warning Transmit Interrupt Mask.<br>The CAN_GIM.EWTIM bit enables (unmasks) the error warning transmit interrupt. |
|         |          | 0 Disable Interrupt (Mask)                                                                                              |
|         |          | 1 Enable Interrupt (Unmask)                                                                                             |

## Global CAN Interrupt Flag Register

The CAN\_GIF register, CAN\_GIF register, and CAN\_GIM register control CAN interrupts. For detailed information about interrupt operations, see the Event Control section.



The CAN\_GIF register holds the interrupt flag. The CAN\_INT.GIRQ bit is only asserted if a bit in the CAN\_GIF is set. The CAN\_INT.GIRQ bit remains set as long as at least one bit in the CAN\_GIF register is set. All bits in this register are W1C.

**CAN\_GIF: Global CAN Interrupt Flag Register - R/NW**

Reset = 0x0000

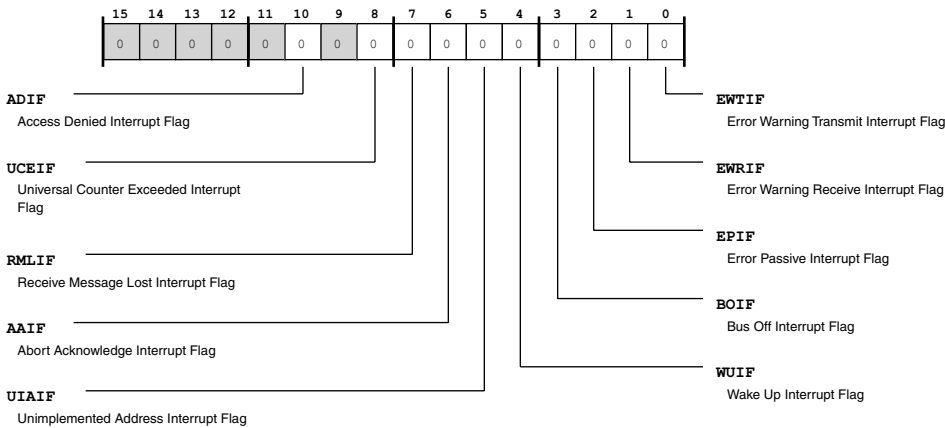


**Figure 21-43: CAN\_GIF Register Diagram**

**Table 21-41: CAN\_GIF Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | ADIF     | Access Denied Interrupt Flag.<br>The CAN_GIF.ADIF bit indicates the access denied interrupt flag is set (latched).                            |
|         |          | 0 No Interrupt Flag                                                                                                                           |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                                |
| 8       | UCEIF    | Universal Counter Exceeded Interrupt Flag.<br>The CAN_GIF.UCEIF bit indicates the universal counter exceeded interrupt flag is set (latched). |
|         |          | 0 No Interrupt Flag                                                                                                                           |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                                |

**Table 21-41:** CAN\_GIF Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| 7       | RMLIF    | Receive Message Lost Interrupt Flag.<br>The CAN_GIF.RMLIF bit indicates the receive message lost interrupt flag is set (latched).   |
|         |          | 0 No Interrupt Flag                                                                                                                 |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                      |
| 6       | AAIF     | Abort Acknowledge Interrupt Flag.<br>The CAN_GIF.AAIF bit indicates the abort acknowledge interrupt flag is set (latched).          |
|         |          | 0 No Interrupt Flag                                                                                                                 |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                      |
| 5       | UIAIF    | Unimplemented Address Interrupt Flag.<br>The CAN_GIF.UIAIF bit indicates the unimplemented address interrupt flag is set (latched). |
|         |          | 0 No Interrupt Flag                                                                                                                 |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                      |
| 4       | WUIF     | Wake Up Interrupt Flag.<br>The CAN_GIF.WUIF bit indicates the wake up interrupt flag is set (latched).                              |
|         |          | 0 No Interrupt Flag                                                                                                                 |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                      |
| 3       | BOIF     | Bus Off Interrupt Flag.<br>The CAN_GIF.BOIF bit indicates the bus off interrupt flag is set (latched).                              |
|         |          | 0 No Interrupt Flag                                                                                                                 |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                      |

**Table 21-41:** CAN\_GIF Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | EPIF     | Error Passive Interrupt Flag.<br>The <code>CAN_GIF.EPIF</code> bit indicates the error passive mode interrupt flag is set (latched).               |
|         |          | 0 No Interrupt Flag                                                                                                                                |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                                     |
| 1       | EWRIF    | Error Warning Receive Interrupt Flag.<br>The <code>CAN_GIF.EWRIF</code> bit indicates the error warning receive interrupt flag is set (latched).   |
|         |          | 0 No Interrupt Flag                                                                                                                                |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                                     |
| 0       | EWTIF    | Error Warning Transmit Interrupt Flag.<br>The <code>CAN_GIF.EWTIF</code> bit indicates the error warning transmit interrupt flag is set (latched). |
|         |          | 0 No Interrupt Flag                                                                                                                                |
|         |          | 1 Interrupt Flag Set (Latched)                                                                                                                     |

## CAN Master Control Register

The `CAN_CTL` register controls CAN mode requests, including soft reset.

### CAN\_CTL: CAN Master Control Register - R/W

Reset = 0x0080

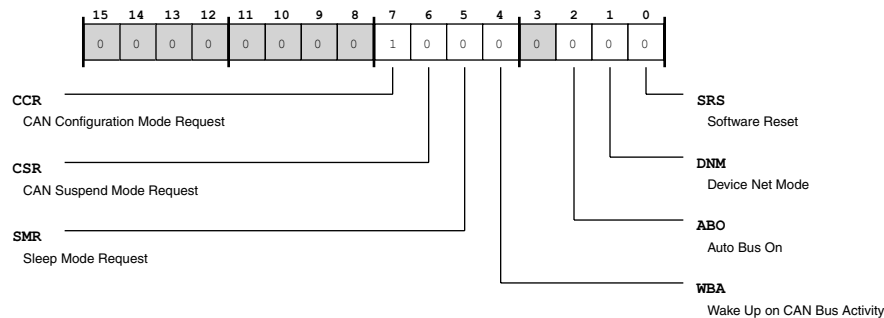


**Figure 21-44:** CAN\_CTL Register Diagram

**Table 21-42:** CAN\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | CCR      | CAN Configuration Mode Request.<br>The CAN_CTL.CCR bit requests that the CAN enter configuration mode. Note that the CAN should always be put in configuration mode before modifying the CAN_CLK or CAN_TIMING registers.            |
|         |          | 0 No Request (Exit Configuration Mode)                                                                                                                                                                                               |
|         |          | 1 Request Configuration Mode                                                                                                                                                                                                         |
| 6       | CSR      | CAN Suspend Mode Request.<br>The CAN_CTL.CSR bit requests that the CAN enter suspend mode. The CAN enters suspend mode after the current operation of the CAN bus is finished.                                                       |
|         |          | 0 No Request (Exit Suspend Mode)                                                                                                                                                                                                     |
|         |          | 1 Request Suspend Mode                                                                                                                                                                                                               |
| 5       | SMR      | Sleep Mode Request.<br>The CAN_CTL.SMR bit requests that the CAN enter sleep mode. The CAN enters sleep mode after the current operation of the CAN bus is finished.                                                                 |
|         |          | 0 No Request (Exit Sleep Mode)                                                                                                                                                                                                       |
|         |          | 1 Request Sleep Mode                                                                                                                                                                                                                 |
| 4       | WBA      | Wake Up on CAN Bus Activity.<br>The CAN_CTL.WBA bit enables wake on CAN bus activity. When enabled, a dominant bit on the CAN_RX pin ends sleep mode (in addition the default wake up condition of a write to the CAN_INT register). |
|         |          | 0 Disable Wake on Bus Activity                                                                                                                                                                                                       |
|         |          | 1 Enable Wake on Bus Activity                                                                                                                                                                                                        |

**Table 21-42:** CAN\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | ABO      | Auto Bus On.<br>The CAN_CTL.ABO bit selects whether (if enabled) the CAN enters active mode after the BusOff recovery sequence or (if disabled) the CAN enters configuration mode after the BusOff recovery sequence.    |
|         |          | 0      Disable Auto Bus On                                                                                                                                                                                               |
|         |          | 1      Enable Auto Bus On                                                                                                                                                                                                |
| 1       | DNM      | Device Net Mode.<br>The CAN_CTL.DNM bit enables mailbox filtering on a data field. The filtering is done on the standard ID of the message and data fields. For more information, see the CAN_AMnnH.FDF bit description. |
|         |          | 0      Disable Device Net Mode                                                                                                                                                                                           |
|         |          | 1      Enable Device Net Mode                                                                                                                                                                                            |
| 0       | SRS      | Software Reset.<br>The CAN_CTL.SRS bit resets the CAN, bringing all control registers to a defined state. Soft reset is entered immediately after software has set the CAN_CTL.SRS bit.                                  |
|         |          | 0      No Action                                                                                                                                                                                                         |
|         |          | 1      Reset CAN                                                                                                                                                                                                         |

## Interrupt Pending Register

The CAN\_INT register indicates the status of pending CAN interrupts and indicates the state of the CAN\_RX and CAN\_TX pins.

CAN\_INT: Interrupt Pending Register - R/W

Reset = 0x0000

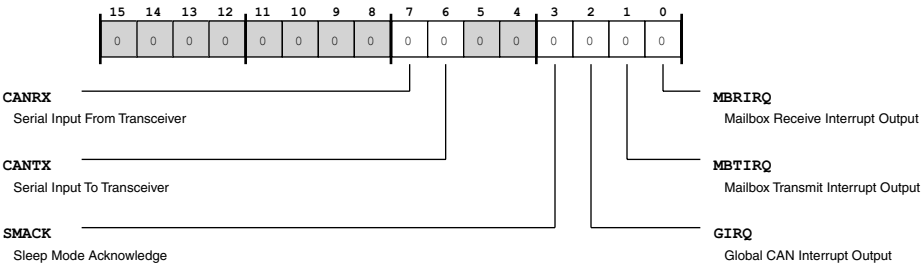


Figure 21-45: CAN\_INT Register Diagram

Table 21-43: CAN\_INT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | CANRX    | Serial Input From Transceiver.<br>The CAN_INT.CANRX bit indicates the logic value that the CAN detects on the CAN_RX pin. Note that the reset/default value for CAN_INT.CANRX is dependent on pin values. |
|         |          | 0 Dominant Value (Low Active)                                                                                                                                                                             |
|         |          | 1 Recessive Value (High Active)                                                                                                                                                                           |
| 6       | CANTX    | Serial Input To Transceiver.<br>The CAN_INT.CANTX bit indicates the logic value that the CAN detects on the CAN_TX pin. Note that the reset/default value for CAN_INT.CANTX is dependent on pin values.   |
|         |          | 0 Dominant Value (Low Active)                                                                                                                                                                             |
|         |          | 1 Recessive Value (High Active)                                                                                                                                                                           |
| 3       | SMACK    | Sleep Mode Acknowledge.<br>The CAN_INT.SMACK bit indicates when the CAN has entered sleep mode.                                                                                                           |
|         |          | 0 Not in Sleep Mode                                                                                                                                                                                       |
|         |          | 1 Sleep Mode                                                                                                                                                                                              |

**Table 21-43:** CAN\_INT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | GIRQ     | Global CAN Interrupt Output.<br>The CAN_INT.GIRQ bit indicates when at least one bit is set in the CAN_GIF register, indicating at least one unmasked CAN is flagged (latched). The CAN_INT.GIRQ bit remains set as long as at least one bit is set in the CAN_GIF register. |
|         |          | 0 No CAN Global Interrupt Flag Set                                                                                                                                                                                                                                           |
|         |          | 1 CAN Global Interrupt Flag (1 or More) Set                                                                                                                                                                                                                                  |
| 1       | MBTIRQ   | Mailbox Transmit Interrupt Output.<br>The CAN_INT.MBTIRQ bit indicates when any bits are set in the CAN_MBTIF1 register or CAN_MBTIF2 register, indicating transmit.                                                                                                         |
|         |          | 0 No CAN Transmit Flags Set                                                                                                                                                                                                                                                  |
|         |          | 1 CAN Transmit Flags Set (1 or More)                                                                                                                                                                                                                                         |
| 0       | MBRIRQ   | Mailbox Receive Interrupt Output.<br>The CAN_INT.MBRIRQ bit indicates when any bits are set in the CAN_MBRIF1 register or CAN_MBRIF2 register, indicating receive.                                                                                                           |
|         |          | 0 No CAN Receive Flags Set                                                                                                                                                                                                                                                   |
|         |          | 1 CAN Receive Flags Set (1 or More)                                                                                                                                                                                                                                          |

## Temporary Mailbox Disable Register

The CAN\_MBTD register supports temporarily and selectively disabling CAN mailboxes. For more information about this feature, see the Operating Modes section.

### CAN\_MBTD: Temporary Mailbox Disable Register - R/W

Reset = 0x0000

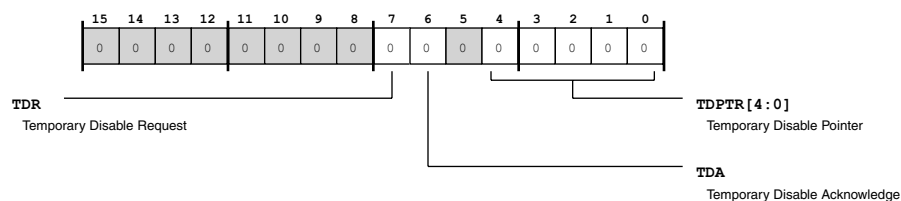


**Figure 21-46:** CAN\_MBTD Register Diagram

**Table 21-44:** CAN\_MBTD Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | TDR      | Temporary Disable Request.<br>The CAN_MBTD.TDR bit hold the pointer to mailbox, which is disabled when the CAN_MBTD.TDR bit is set.                                                                                                                                                                |
|         |          | 0                      No Request                                                                                                                                                                                                                                                                  |
|         |          | 1                      Request Temporary Mailbox Disable                                                                                                                                                                                                                                           |
| 6       | TDA      | Temporary Disable Acknowledge.<br>The CAN_MBTD.TDA bit indicates when the mailbox (to which the CAN_MBTD.TDPTR bit point) is disabled. When this bit is set for a mailbox, only the data field of that mailbox may be updated. Accesses that mailbox's control bits and the identifier are denied. |
|         |          | 0                      No Acknowledge                                                                                                                                                                                                                                                              |
|         |          | 1                      Acknowledge Temporary Mailbox Disable                                                                                                                                                                                                                                       |
| 4:0     | TDPTR    | Temporary Disable Pointer.<br>The CAN_MBTD.TDPTR bits hold the pointer to mailbox, which is disabled when the CAN_MBTD.TDR bit is set.                                                                                                                                                             |

**Error Counter Warning Level Register**

The CAN\_EWR register, CAN\_CEC register, and CAN\_ESR register control CAN warnings and errors. For detailed information about error and warning operations, see the Operating Modes section.

**CAN\_EWR: Error Counter Warning Level Register - R/W**

Reset = 0x0000

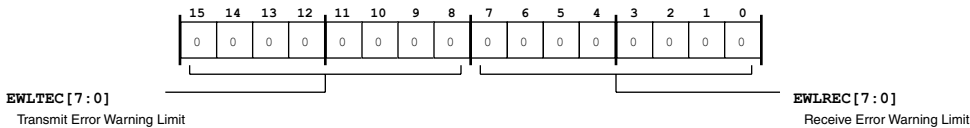


**Figure 21-47:** CAN\_EWR Register Diagram



**Table 21-45:** CAN\_EWR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | EWLTEC   | Transmit Error Warning Limit.<br>The CAN_EWR.EWLTEC bits select the transmit error warning limit, which is used as a condition for the CAN_GIS.EWTIS interrupt. |
| 7:0     | EWLREC   | Receive Error Warning Limit.<br>The CAN_EWR.EWLREC bits select the receive error warning limit, which is used as a condition for the CAN_GIS.EWRIS interrupt.   |

### Error Status Register

The CAN\_ESR register, CAN\_CEC register, and CAN\_EWR register control CAN warnings and errors. All bits in the CAN\_ESR are W1C. Note that the CAN updates the CAN\_CEC register when error status is detected in the CAN\_ESR register. For detailed information about error and warning operations, see the Operating Modes section.

**CAN\_ESR: Error Status Register - R/W**

Reset = 0x0020

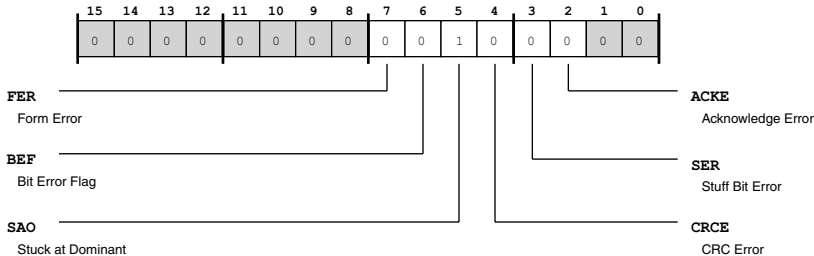


**Figure 21-48:** CAN\_ESR Register Diagram

**Table 21-46:** CAN\_ESR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                   |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 7       | FER      | Form Error.<br>The <code>CAN_ESR.FER</code> bit indicates when a form error occurs, indicating that a fixed-form bit position in the CAN frame contains one or more illegal bits. This occurs when a dominant bit is detected at a delimiter or end-of-frame bit position.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                   |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No Status         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Form Error        |
| 6       | BEF      | Bit Error Flag.<br>The <code>CAN_ESR.BEF</code> bit indicates (detected by the transmitting node only) the value on the <code>CAN_RX</code> pin does not equal what is being transmitted on the <code>CAN_TX</code> pin.<br>When a node is transmitting, it continuously monitors its receive pin ( <code>CAN_RX</code> ) and compares the received data with the transmitted data. The node postpones the transmission (during the arbitration phase) if the received and transmitted data do not match. After the arbitration phase ( <code>CAN_MBnn.ID1.RTR</code> bit sent successfully), a bit error is signaled when the value on the <code>CAN_RX</code> pin does not equal what is being transmitted on the <code>CAN_TX</code> pin. |                   |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No Status         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Bit Error Flag    |
| 5       | SAO      | Stuck at Dominant.<br>The <code>CAN_ESR.SAO</code> bit indicates when the <code>CAN_RX</code> pin sticks at dominant level, indicating that shorted wires are likely.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                   |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | No Status         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Stuck At Dominant |

**Table 21-46:** CAN\_ESR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| 4       | CRCE     | CRC Error.<br>The CAN_ESR.CRCE bit indicates when a CRC error occurs. This error may occur when a receiver calculates the CRC on the data it received and finds the value different than the CRC that was transmitted on the bus.                                                                                                                                                                                                                                                                                                                                         |                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Status               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | CRC Error               |
| 3       | SER      | Stuff Bit Error.<br>The CAN_ESR.SER bit indicates when a stuff bit error (stuffed 6th consecutive bit value is the same as the previous five bits) occurs. The CAN specification requires that the transmitter insert an extra stuff bit of opposite value after 5 bits have been transmitted with the same value. The receiver disregards the value of these stuff bits. The receiver takes advantage of the signal edge to re-synchronize itself. A stuff bit error occurs on receiving nodes when the 6th consecutive bit value is the same as the previous five bits. |                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Status               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Stuff Bit Error Receive |
| 2       | ACKE     | Acknowledge Error.<br>The CAN_ESR.ACKE bit indicates when an acknowledge error occurs, indicating that a message is sent and no receivers drive an acknowledge bit.                                                                                                                                                                                                                                                                                                                                                                                                       |                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Status               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Acknowledge Error       |

## Universal Counter Register

The `CAN_UCCNT` register holds the current universal count. This register is re-loaded from the `CAN_UCRC` register when the decrements to zero in auto-transmit mode.

CAN\_UCCNT: Universal Counter Register - R/W

Reset = 0x0000

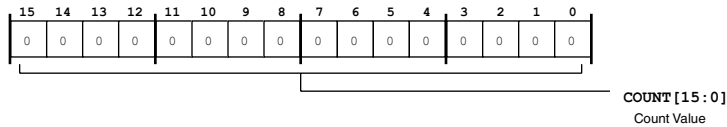


Figure 21-49: CAN\_UCCNT Register Diagram

Table 21-47: CAN\_UCCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                          |
|---------|----------|----------------------------------------------------------------------------------|
| 15:0    | COUNT    | Count Value.<br>The CAN_UCCNT.COUNT bits hold the current universal count value. |

Universal Counter Reload/Capture Register

The CAN\_UCRC register holds the period value (universal count), which is used in auto-transmit mode as the period for sending the message in mailbox 11 (broadcast heartbeat) to all CAN nodes. Accordingly, messages sent this way usually have high priority.

The period value is written to the CAN\_UCRC register. When auto-transmit mode is enabled (CAN\_UCCNF.UCCNF = 0x3), the CAN loads the counter with the value in CAN\_UCRC. The counter decrements to 0 at the CAN bit clock rate, then is reloaded. Each time the counter decrements to 0, the CAN sets the CAN\_TRS1.MB bit for mailbox 11 and sends the corresponding message from mailbox 11.

Note that for auto-transmit mode, mailbox 11 must be configured as a transmit mailbox and must contain valid data (identifier, control bits, and data). This setup must occur before the counter first expires after this mode is enabled.

CAN\_UCRC: Universal Counter Reload/Capture Register - R/W

Reset = 0x0000

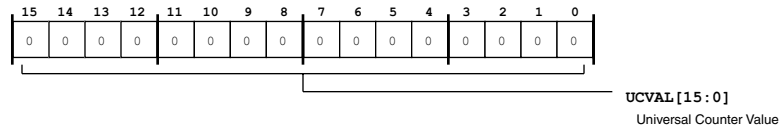


Figure 21-50: CAN\_UCRC Register Diagram

**Table 21-48:** CAN\_UCRC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | UCVAL    | Universal Counter Value.<br>The <code>CAN_UCRC.UCVAL</code> bits hold the value for the universal count period, which is used in auto-transmit mode. |

### Universal Counter Configuration Mode Register

The `CAN_UCCNF` register controls the operation of the universal counter, including counter enable and counter mode selection.

**CAN\_UCCNF: Universal Counter Configuration Mode Register - R/W**

Reset = 0x0000

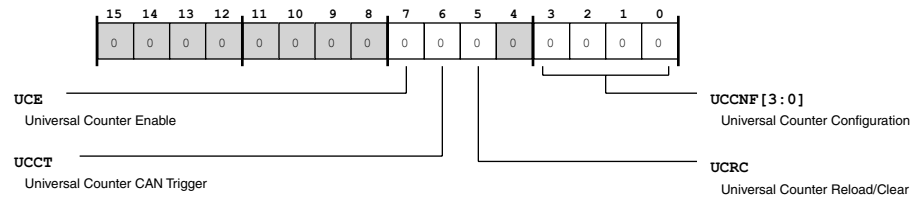


**Figure 21-51:** CAN\_UCCNF Register Diagram

**Table 21-49:** CAN\_UCCNF Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | UCE      | Universal Counter Enable.<br>The <code>CAN_UCCNF.UCE</code> bit enables universal counter operation in the mode selected by the <code>CAN_UCCNF.UCCNF</code> bits. |
|         |          | 0      Disable Counter                                                                                                                                             |
|         |          | 1      Enable Counter                                                                                                                                              |

**Table 21-49:** CAN\_UCCNF Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | UCCT     | Universal Counter CAN Trigger.<br>The CAN_UCCNF.UCCT bit enables the universal counter trigger, directing the CAN to re-load the counter on mailbox 4 reception in watchdog mode and clear the counter on mailbox 4 reception in time stamp mode. This bit has no effect in all other modes. |
|         |          | 0      Disable Trigger                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable Trigger                                                                                                                                                                                                                                                                        |
| 5       | UCRC     | Universal Counter Reload/Clear.<br>The CAN_UCCNF.UCRC bit re-loads or clears the universal counter, depending on the counter mode. In watchdog mode, setting this bit directs the CAN to re-load the counter. In all other modes, setting this bit directs the CAN to clear the counter.     |
|         |          | 0      No Action                                                                                                                                                                                                                                                                             |
|         |          | 1      Re-load or Clear the Counter                                                                                                                                                                                                                                                          |

**Table 21-49:** CAN\_UCCNF Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | UCCNF    | Universal Counter Configuration.<br>The CAN_UCCNF.UCCNF bits select the universal counter operating mode. For more information about these modes, see the Operating Modes section. |
|         |          | 0 Reserved                                                                                                                                                                         |
|         |          | 1 Time Stamp Mode                                                                                                                                                                  |
|         |          | 2 Watchdog Mode                                                                                                                                                                    |
|         |          | 3 Auto-transmit Mode                                                                                                                                                               |
|         |          | 4 Reserved                                                                                                                                                                         |
|         |          | 5 Reserved                                                                                                                                                                         |
|         |          | 6 Count Error Frames                                                                                                                                                               |
|         |          | 7 Count Overload Frames                                                                                                                                                            |
|         |          | 8 Count Arbitration Lost                                                                                                                                                           |
|         |          | 9 Count Aborted Transmissions                                                                                                                                                      |
|         |          | 10 Count Successful Transmissions                                                                                                                                                  |
|         |          | 11 Count Rejected Receive Messages                                                                                                                                                 |
|         |          | 12 Count Receive Message Lost                                                                                                                                                      |
|         |          | 13 Count Successful Receptions                                                                                                                                                     |
|         |          | 14 Count Stored Receptions                                                                                                                                                         |
|         |          | 15 Count Valid Messages                                                                                                                                                            |

## Acceptance Mask (L) Register

The CAN\_AMnnL register and CAN\_AMnnH register manage acceptance mask operation. For information about acceptance mask operation, see the Receive Operation section.

CAN\_AMnnL: Acceptance Mask (L) Register - R/W

Reset = 0x0000

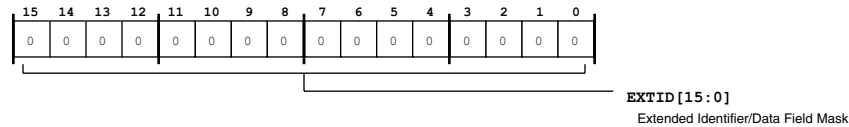


Figure 21-52: CAN\_AMnnL Register Diagram

Table 21-50: CAN\_AMnnL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | EXTID    | Extended Identifier/Data Field Mask.<br>The CAN_AMnnL.EXTID bits hold the extended ID (high bits) or data field mask for acceptance mask operations. |

Acceptance Mask (H) Register

The CAN\_AMnnH register and CAN\_AMnnL register manage acceptance mask operation. For information about acceptance mask operation, see the Receive Operation section.

CAN\_AMnnH: Acceptance Mask (H) Register - R/W

Reset = 0x0000

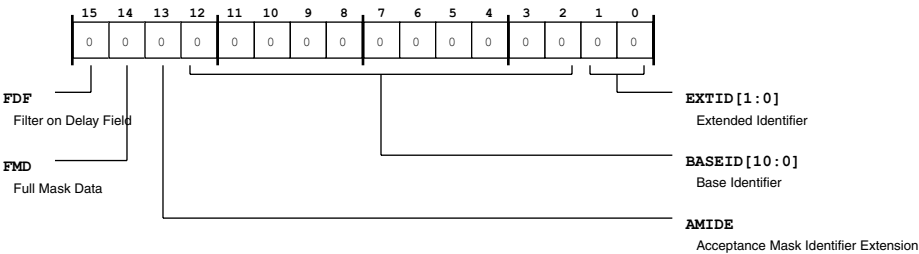


Figure 21-53: CAN\_AMnnH Register Diagram



**Table 21-51:** CAN\_AMnnH Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | FDF      | Filter on Delay Field.<br>The CAN_AMnnH.FDF bit selects the operation of the CAN_AMnnH register and CAN_AMnnL register when the CAN_CTL.DNM bit is enabled.<br>If the CAN_AMnnH.FDF bit is set, the corresponding CAN_AMnnL.EXTID bits hold the data field mask. If the CAN_AMnnH.FDF bit is cleared, the corresponding CAN_AMnnL.EXTID bits hold the high bits of the extended identifier mask. |
| 14      | FMD      | Full Mask Data.<br>The CAN_AMnnH.FMD bit works with the CAN_AMnnH.FDF bit to determine data field filtering. For information about data field filtering, see the Receive Operation section.                                                                                                                                                                                                      |
| 13      | AMIDE    | Acceptance Mask Identifier Extension.<br>The CAN_AMnnH.AMIDE bit enables the comparison of the received message ID to the value in the CAN_AMnnH.EXTID and CAN_AMnnL.EXTID bits.                                                                                                                                                                                                                 |
| 12:2    | BASEID   | Base Identifier.<br>The CAN_AMnnH.BASEID bits hold the base ID for acceptance mask operations.                                                                                                                                                                                                                                                                                                   |
| 1:0     | EXTID    | Extended Identifier.<br>The CAN_AMnnH.EXTID bits hold the extended ID (low bits) for acceptance mask operations.                                                                                                                                                                                                                                                                                 |

## Mailbox Word 0 Register

The CAN\_MBnn\_DATA0 register holds mailbox data bytes.

**CAN\_MBnn\_DATA0: Mailbox Word 0 Register - R/W**

Reset = 0x0000

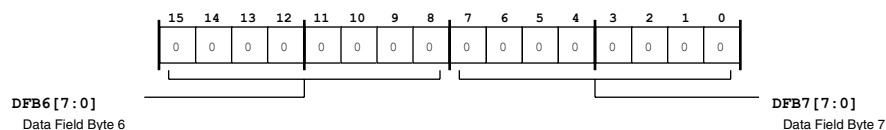


**Figure 21-54:** CAN\_MBnn\_DATA0 Register Diagram

Table 21-52: CAN\_MBnn\_DATA0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                               |
|---------|----------|-----------------------------------------------------------------------|
| 15:8    | DFB6     | Data Field Byte 6.<br>The CAN_MBnn_DATA0.DFB6 bits hold mailbox data. |
| 7:0     | DFB7     | Data Field Byte 7.<br>The CAN_MBnn_DATA0.DFB7 bits hold mailbox data. |

Mailbox Word 1 Register

The CAN\_MBnn\_DATA1 register holds mailbox data bytes.

CAN\_MBnn\_DATA1: Mailbox Word 1 Register - R/W

Reset = 0x0000

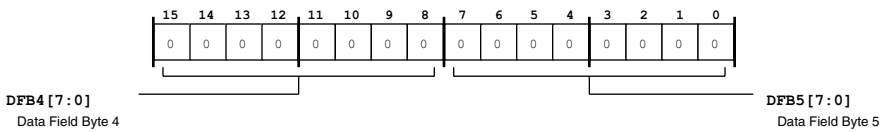


Figure 21-55: CAN\_MBnn\_DATA1 Register Diagram

Table 21-53: CAN\_MBnn\_DATA1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                               |
|---------|----------|-----------------------------------------------------------------------|
| 15:8    | DFB4     | Data Field Byte 4.<br>The CAN_MBnn_DATA1.DFB4 bits hold mailbox data. |
| 7:0     | DFB5     | Data Field Byte 5.<br>The CAN_MBnn_DATA1.DFB5 bits hold mailbox data. |

Mailbox Word 2 Register

The CAN\_MBnn\_DATA2 register holds mailbox data bytes.

**CAN\_MBnn\_DATA2: Mailbox Word 2 Register - R/W**

Reset = 0x0000

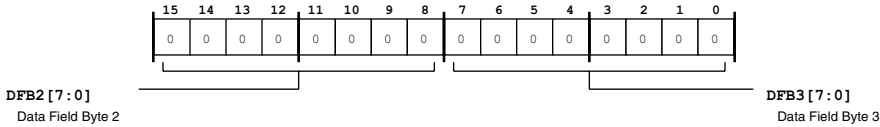


**Figure 21-56:** CAN\_MBnn\_DATA2 Register Diagram

**Table 21-54:** CAN\_MBnn\_DATA2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                               |
|---------|----------|-----------------------------------------------------------------------|
| 15:8    | DFB2     | Data Field Byte 2.<br>The CAN_MBnn_DATA2.DFB2 bits hold mailbox data. |
| 7:0     | DFB3     | Data Field Byte 3.<br>The CAN_MBnn_DATA2.DFB3 bits hold mailbox data. |

Mailbox Word 3 Register

The CAN\_MBnn\_DATA3 register holds mailbox data bytes.

**CAN\_MBnn\_DATA3: Mailbox Word 3 Register - R/W**

Reset = 0x0000

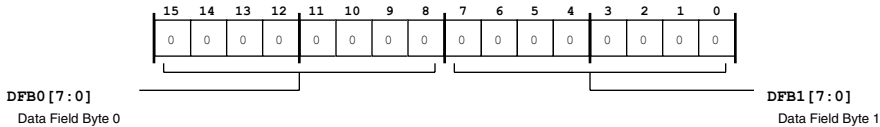


**Figure 21-57:** CAN\_MBnn\_DATA3 Register Diagram

**Table 21-55:** CAN\_MBnn\_DATA3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                               |
|---------|----------|-----------------------------------------------------------------------|
| 15:8    | DFB0     | Data Field Byte 0.<br>The CAN_MBnn_DATA3.DFB0 bits hold mailbox data. |
| 7:0     | DFB1     | Data Field Byte 1.<br>The CAN_MBnn_DATA3.DFB1 bits hold mailbox data. |

## Mailbox Length Register

The CAN\_MBnn\_LENGTH register holds the data length code for the received remote frame. For more information about remote frames, see the Remote Frame Handling section.

CAN\_MBnn\_LENGTH: Mailbox Length Register - R/W

Reset = 0x0000

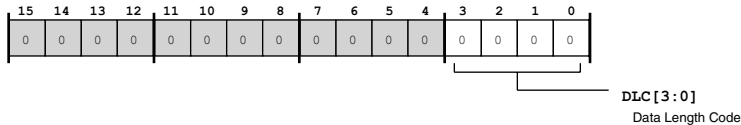


Figure 21-58: CAN\_MBnn\_LENGTH Register Diagram

Table 21-56: CAN\_MBnn\_LENGTH Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | DLC      | Data Length Code.<br>The CAN_MBnn_LENGTH.DLC bits hold the DLC value of the received remote frame. The received value overwrites any previous value. |

## Mailbox Timestamp Register

The CAN\_MBnn\_TIMESTAMP register holds an indication of the time of reception or transmission for each message, when the universal counter is in time stamp mode (CAN\_UCCNF.UCCNF =0x1). In this mode, the CAN writes the value of the counter (CAN\_UCCNT) to the CAN\_MBnn\_TIMESTAMP register when a received message is stored or a message is transmitted. For more information about timestamps, see the Time Stamps section.

CAN\_MBnn\_TIMESTAMP: Mailbox Timestamp Register - R/W

Reset = 0x0000

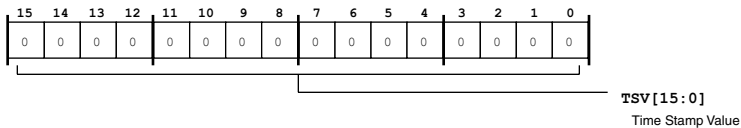


Figure 21-59: CAN\_MBnn\_TIMESTAMP Register Diagram

**Table 21-57:** CAN\_MBnn\_TIMESTAMP Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                |
|---------|----------|----------------------------------------------------------------------------------------|
| 15:0    | TSV      | Time Stamp Value.<br>The CAN_MBnn_TIMESTAMP.TSV bits hold the message timestamp value. |

## Mailbox ID 0 Register

The CAN\_MBnn\_ID0 register and CAN\_MBnn\_ID1 register manage acceptance mask operations. For information about acceptance mask operation, see the Receive Operation section.

**CAN\_MBnn\_ID0: Mailbox ID 0 Register - R/W**

Reset = 0x0000

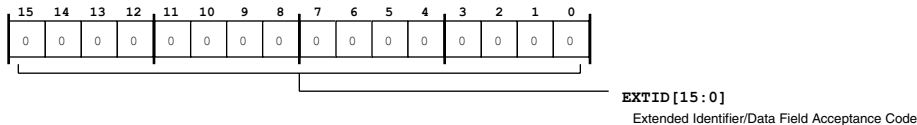


**Figure 21-60:** CAN\_MBnn\_ID0 Register Diagram

**Table 21-58:** CAN\_MBnn\_ID0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | EXTID    | Extended Identifier/Data Field Acceptance Code.<br>The CAN_MBnn_ID0.EXTID bits hold the extended ID (high bits) or data field mask for acceptance mask operations. |

## Mailbox ID 1 Register

The CAN\_MBnn\_ID1 register and CAN\_MBnn\_ID0 register manage acceptance mask operations. For information about acceptance mask operation, see the Receive Operation section.

**CAN\_MBnn\_ID1: Mailbox ID 1 Register - R/W**

Reset = 0x0000

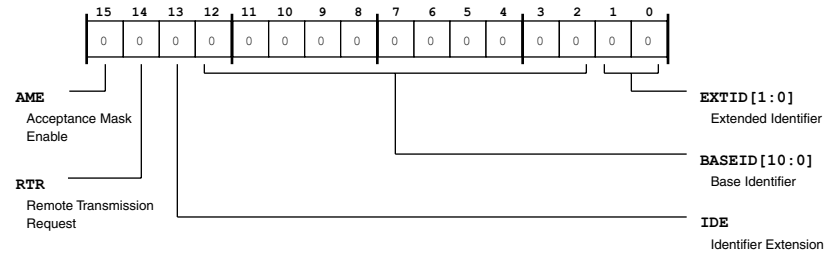


**Figure 21-61:** CAN\_MBnn\_ID1 Register Diagram

**Table 21-59:** CAN\_MBnn\_ID1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AME      | Acceptance Mask Enable.<br>The CAN_MBnn_ID1.AME bit enables acceptance mask operations. When enabled (=1), only those bits that have the corresponding mask bit cleared are compared to the received message ID. A bit position that is set in the mask register does not need to match. |
| 14      | RTR      | Remote Transmission Request.<br>The CAN_MBnn_ID1.RTR bit selects whether the frame contains data (data frame) or contains a request for data associated with the message identifier in the frame being sent (remote frame).                                                              |
| 13      | IDE      | Identifier Extension.<br>The CAN_MBnn_ID1.IDE bit enables the comparison of the received message ID to the value in the CAN_MBnn_ID1.EXTID and CAN_MBnn_ID0.EXTID bits.                                                                                                                  |
| 12:2    | BASEID   | Base Identifier.<br>The CAN_MBnn_ID1.BASEID bits hold the base ID for acceptance mask operations.                                                                                                                                                                                        |
| 1:0     | EXTID    | Extended Identifier.<br>The CAN_MBnn_ID1.EXTID bits ld the extended ID (low bits) for acceptance mask operations.                                                                                                                                                                        |

## 22 Universal Serial Bus (USB)

The USB OTG controller provides a low-cost connectivity solution for consumer mobile devices such as cell phones, digital still cameras and MP3 players, allowing these devices to transfer data using a point-to-point USB connection without the need for a personal computer host.

The USB controller can operate in a traditional USB peripheral-only mode as well as the host mode presented in the On-The-Go (OTG) supplement <sup>1</sup> to the USB 2.0 Specification<sup>2</sup>In host mode, the USB module supports transfers at high-speed (480Mbps), full-speed (12Mbps), and low-speed (1.5 Mbp/sec) rates. Peripheral mode supports the high- and full-speed transfer rates.

The USB controller uses a peripheral bus slave interface to access its control and status registers as well as read and write to the endpoint packet buffers. Data is transferred to and from the USB controller through any of the 11 transmit and 11 receive endpoint FIFOs, EP1 - EP11, providing a total of 22 data endpoints.

### USB Features

The USB controller provides the following features:

- Low speed, full speed, high speed rates supported
- One bidirectional control endpoint
- Eleven transmit and eleven receive unidirectional endpoints
- 16 KB dynamically configured FIFO RAM
- Eight DMA master channels
- Two top-level maskable general purpose interrupts
- Low power wakeup on activity
- VBUS control interrupts for external analog VBUS control
- Session request protocol (SRP) and host negotiation protocol (HNP) capability
- Host transaction scheduling in hardware
- Soft connect/disconnect feature
- Full- and high-speed physical layer UTMI+ level 3 interface for on-chip PHY

---

1.On-The-Go Supplement to the USB 2.0 Specification, Rev 1.0a; June 24, 2003; USB-IF

2.Universal Serial Bus Specification 2.0

- Backwards compatible with existing USB 1.1 hosts
- Support for Battery Charging Specification Revision 1.1

The number of active endpoints at one time is only limited by device requirements or system bandwidth, because each endpoint operates independently from the next. Software determines the type of transfer for each endpoint individually and also the manner in which it is transferred between the USB controller and memory (DMA or interrupt-based). Endpoint zero is used solely for receive and transmit control transfers, which are used for device configuration and information gathering.

## USB Functional Description

The following sections describe the function of the USB OTG interface.

### USB Interface Overview

The USB controller operates in either of two USB operation modes (peripheral or host mode) at a given time.

In peripheral mode, the USB controller encodes, decodes, checks, and directs all USB packets sent and received, responding appropriately to host requests. Data is transferred from the processor core memory into the device's TX FIFOs to be transmitted onto USB as IN packets. In the other direction USB OUT packets are received into the RX FIFOs (having been sent from the host) and transferred to system memory for processing or storage. In peripheral mode, the USB controller acts as a slave device to another USB host; either a personal computer or another OTG host controller.

When operating in host mode, the USB controller uses simple hosting capabilities to master point-to-point connections with another USB peripheral, initiating transfers on the bus for the peripheral to respond. USB IN packets are received into the RX FIFOs to be moved into the processor core memory, and data written into TX FIFOs is transmitted onto the bus as USB OUT packets. In this mode, the USB controller encodes, decodes, and checks USB packets sent and received. The controller automatically schedules isochronous and interrupt transfers from the endpoint buffers such that one transaction is performed every  $n$  frames, where  $n$  represents the polling interval programmed for the endpoint.

Any of the endpoints can be programmed to be written to or read from using the DMA master channels to provide the most efficient means of transferring data between the controller and on-chip memory. USB endpoints 0 through 11 have DMA interrupt lines (USB\_DMAxINT) providing a total of eight DMA request lines. Two top-level maskable interrupts are provided, each of which can be sourced from any or all of transmit endpoint status, receive endpoint status or global USB status. Details of these can be found in [USB Interrupt Signals](#).

The USB controller uses the slave bus to access control and status registers and FIFOs from a slave perspective and to transfer data between the USB engine and on-chip memory as a master. The MMR peripheral data bus is 16-bits wide, the DMA data bus is 32-bits wide.



The USB controller's RAM interface supports a single block of synchronous single-port RAM used to buffer the USB packets. 16K bytes of SRAM are available.

The UTMI+ level 3 PHY interface provides a means of connecting a selection of high- or full-speed PHYs to the controller, from device-only PHYs through full OTG compliant PHYs. The details of the PHY interface can be found in [USB UTMI Interface](#).

**ATTENTION:** Check the processor data sheet for requirements regarding minimum system clock frequency needed for proper USB operation.

The USB controller is configured as either a USB OTG A device or B device depending on the type of plug inserted into its USB receptacle. This is determined by the state of the `USB_ID` (connector ID) pin.

The asynchronous wakeup circuit is used to detect when another 'B' device is asserting its D+ pull-up to initiate the SRP (session request protocol) when all other clocks are off. This circuit requires a slow clock (for example, 32kHz). This slow clock is derived from `SCLK` and enabled in the USB SRP clock divider register.

Use of the controller for OTG functionality requires the capability to drive `VBUS` (as a default A device powering the bus), to discharge `VBUS` (speeding up the time for `VBUS` to fall below the `SessionEnd` threshold as a 'B' device checking initial conditions), and to charge `VBUS` to 2.1V (when initiating SRP as a B device). These controls are driven from the UTMI interface, but the controller also provides a separate interrupt register, `USB_OTG_VBUS_IRQ`, which represents the drive `VBUS`, discharge `VBUS`, and charge `VBUS` signaling. See the `USB_OTG_VBUS_IRQ` register section for more information on these controls.

## Multi-point Support

The USB controller has the facility, when operating in Host mode, to act as the host to a range of USB peripheral devices – high-speed, full-speed or low-speed – where these devices are connected to the USB controller via a USB hub. The key feature of the controller's support for multiple devices is its facility to allow the functions of the target devices to be individually allocated to the different Rx and Tx endpoints implemented. Furthermore, this allocation can be made dynamically, allowing the devices from the targeted peripheral list to be used in different combinations. The combinations of peripheral devices that may be used together are limited by the numbers of Tx and Rx endpoints implemented in the controller. Further devices can only be added where the endpoints they require remain available.

## USB On-Chip Bus Interfaces

The USB controller uses two 32-bit wide independent bus interfaces, a master and a slave, to communicate with a processor-based subsystem. The slave interface allows the processor core to access the control and status registers (including DMA master registers) and the endpoint FIFOs. The master interface is used by the integrated DMA to drive data into or out of the endpoint FIFOs with minimal processor core interaction.

## USB FIFO Configuration

Each bidirectional endpoint (provided as two unidirectional endpoints) has its own endpoint number (0 for control, 1-11 for data transfer). Although two endpoints might use the same number, the endpoints may support different transfer types. Each of these bidirectional endpoints has a fixed region of the SRAM in the USB controller to which it has access, and this feature dictates to some extent the types of transfers that may be used for that particular endpoint. This restriction follows from the maximum size of USB packets, which varies with each transfer type. *FIFO Sizes and Transfer Types* lists the endpoint FIFO configuration, with an indication of the transfer types possible for that particular buffer size.

**Table 22-1:** FIFO Sizes and Transfer Types

| Bidirectional Endpoint<br>(RX and TX) | FIFO Size<br>(each direction)                              | USB Transfer Types               |
|---------------------------------------|------------------------------------------------------------|----------------------------------|
| 0                                     | 64 bytes                                                   | Size fixed for control transfers |
| 1-11                                  | Dynamically configured in powers of 2 from 8 to 8192 bytes | Bulk, Interrupt, Isochronous     |

Each endpoint FIFO can buffer one or two packets (in double-buffered mode). Double-buffering is recommended for most applications to improve efficiency by reducing the frequency with which each endpoint needs to be serviced. Double-buffering Bulk transactions means that data transfer over the USB is not slowed if packets can be loaded/unloaded from the FIFO in the time it takes to transfer a packet over the bus. Double-buffering isochronous transactions also allows more time to load/unload the FIFO, but in addition, it also allows the SOF interrupt to be used to service the endpoint rather than the endpoint interrupt. This has the following advantages:

- Easy detection of lost packets
- Regular interrupt timing (making it easier to source/sink the data)
- If more than one Isochronous endpoint is used, they can all be serviced with one interrupt.

## USB Power and Clocking

The USB controller uses the system clock `SCLK` to generate an internal clock used to clock the USB registers. The transceiver clock is a 60MHz clock sourced from the UTMI PHY and is used by the PHY interface logic and USB engine. A 32 KHz clock (refer to the `USB_SRP_CLKDIV` register) is used for D+ pulse detection for SRP signaling by an OTG B device only.

During SUSPEND and when no session is active, the clock to much of the USB controller is stopped to reduce power consumption. The clock becomes operational again when RESUME signaling is detected on the USB lines.

## USB UTMI Interface

The interface to the on-chip PHY uses the industry-standard UTMI+ (universal transceiver macro interface) level 3. This provides full high-speed device and OTG functionality and supports communication to a hub.

The PHY is a mixed-signal block and includes the following:

- Full-speed and high-speed drivers and receivers (single-ended and differential)
- Full-speed and high-speed CDR
- Full-speed/high-speed shift registers, NRZI encode/decode and bit-stuff encode/decode
- Data line pull-up and pull-down resistors
- VBUS and USB\_ID level detection
- Host disconnect detection

Although the UTMI specification indicates that VBUS charging, driving and discharging be done inside the PHY, for process-restricting and power reasons, these functions need to be implemented off-chip in a separate USB charge-pump chip.

## USB Register List

The universal serial bus (USB) controller is a multipoint high-speed dual role USB 2.0 compliant controller. The USB controller can operate in a traditional USB peripheral-only mode as well as the host mode presented in the on-the-go (OTG) supplement to the USB 2.0 Specification, Rev 1.0a; June 24, 2003; USB-IF. A set of registers govern USB controller operations. For more information on USB controller functionality, see the USB controller register descriptions.

**Table 22-2:** USB Module Registers

| Name        | Description                        |
|-------------|------------------------------------|
| USB_FADDR   | Function Address Register          |
| USB_POWER   | Power and Device Control Register  |
| USB_INTRTX  | Transmit Interrupt Register        |
| USB_INTRRX  | Receive Interrupt Register         |
| USB_INTRTXE | Transmit Interrupt Enable Register |
| USB_INTRRXE | Receive Interrupt Enable Register  |

**Table 22-2:** USB Module Registers (Continued)

| Name               | Description                            |
|--------------------|----------------------------------------|
| USB_IRQ            | Common Interrupts Register             |
| USB_IEN            | Common Interrupts Enable Register      |
| USB_FRAME          | Frame Number Register                  |
| USB_INDEX          | Index Register                         |
| USB_TESTMODE       | Testmode Register                      |
| USB_FIFOBn         | FIFO Byte (8-Bit) Register             |
| USB_FIFOHn         | FIFO Half-Word (16-Bit) Register       |
| USB_FIFOn          | FIFO Word (32-Bit) Register            |
| USB_DEV_CTL        | Device Control Register                |
| USB_TXFIFOSZ       | Transmit FIFO Size Register            |
| USB_RXFIFOSZ       | Receive FIFO Size Register             |
| USB_TXFIFOADDR     | Transmit FIFO Address Register         |
| USB_RXFIFOADDR     | Receive FIFO Address Register          |
| USB_EPINFO         | Endpoint Information Register          |
| USB_RAMINFO        | RAM Information Register               |
| USB_LINKINFO       | Link Information Register              |
| USB_VPLEN          | VBUS Pulse Length Register             |
| USB_HS_EOF1        | High-Speed EOF 1 Register              |
| USB_FS_EOF1        | Full-Speed EOF 1 Register              |
| USB_LS_EOF1        | Low-Speed EOF 1 Register               |
| USB_SOFT_RST       | Software Reset Register                |
| USB_MPN_TXFUNCADDR | MPn Transmit Function Address Register |

**Table 22-2:** USB Module Registers (Continued)

| Name               | Description                                                 |
|--------------------|-------------------------------------------------------------|
| USB_MPn_TXHUBADDR  | MPn Transmit Hub Address Register                           |
| USB_MPn_TXHUBPORT  | MPn Transmit Hub Port Register                              |
| USB_MPn_RXFUNCADDR | MPn Receive Function Address Register                       |
| USB_MPn_RXHUBADDR  | MPn Receive Hub Address Register                            |
| USB_MPn_RXHUBPORT  | MPn Receive Hub Port Register                               |
| USB_EPn_TXMAXP     | EPn Transmit Maximum Packet Length Register                 |
| USB_EP0_CSRn_H     | EP0 Configuration and Status (Host) Register                |
| USB_EP0_CSRn_P     | EP0 Configuration and Status (Peripheral) Register          |
| USB_EPn_TXCSR_H    | EPn Transmit Configuration and Status (Host) Register       |
| USB_EPn_TXCSR_P    | EPn Transmit Configuration and Status (Peripheral) Register |
| USB_EPn_RXMAXP     | EPn Receive Maximum Packet Length Register                  |
| USB_EPn_RXCSR_H    | EPn Receive Configuration and Status (Host) Register        |
| USB_EPn_RXCSR_P    | EPn Receive Configuration and Status (Peripheral) Register  |
| USB_EP0_CNTn       | EP0 Number of Received Bytes Register                       |
| USB_EPn_RXCNT      | EPn Number of Bytes Received Register                       |
| USB_EPn_TXTYPE     | EPn Transmit Type Register                                  |
| USB_EP0_TYPERn     | EP0 Connection Type Register                                |
| USB_EP0_NAKLIMITn  | EP0 NAK Limit Register                                      |
| USB_EPn_TXINTERVAL | EPn Transmit Polling Interval Register                      |
| USB_EPn_RXTYPE     | EPn Receive Type Register                                   |
| USB_EPn_RXINTERVAL | EPn Receive Polling Interval Register                       |
| USB_EP0_CFGDATAn   | EP0 Configuration Information Register                      |

**Table 22-2:** USB Module Registers (Continued)

| Name                       | Description                                   |
|----------------------------|-----------------------------------------------|
| USB_DMA_IRQ                | DMA Interrupt Register                        |
| USB_DMA <sub>n</sub> _CTL  | DMA Channel n Control Register                |
| USB_DMA <sub>n</sub> _ADDR | DMA Channel n Address Register                |
| USB_DMA <sub>n</sub> _CNT  | DMA Channel n Count Register                  |
| USB_RQPKTCNT <sub>n</sub>  | EP <sub>n</sub> Request Packet Count Register |
| USB_CT_UCH                 | Chirp Timeout Register                        |
| USB_CT_HHSRTN              | Host High Speed Return to Normal Register     |
| USB_CT_HSBT                | High Speed Timeout Register                   |
| USB_LPM_ATTR               | LPM Attribute Register                        |
| USB_LPM_CTL                | LPM Control Register                          |
| USB_LPM_IEN                | LPM Interrupt Enable Register                 |
| USB_LPM_IRQ                | LPM Interrupt Status Register                 |
| USB_LPM_FADDR              | LPM Function Address Register                 |
| USB_VBUS_CTL               | VBUS Control Register                         |
| USB_BAT_CHG                | Battery Charging Control Register             |
| USB_PHY_CTL                | PHY Control Register                          |
| USB_PLL_OSC                | PLL and Oscillator Control Register           |

## ADSP-BF60x USB Interrupt List

**Table 22-3:** ADSP-BF60x USB Interrupt List

| Description                 | ID  | DMA Channel | Sensitivity |
|-----------------------------|-----|-------------|-------------|
| USB0 Status/FIFO Data Ready | 122 |             | LEVEL       |

**Table 22-3:** ADSP-BF60x USB Interrupt List (Continued)

| Description                       | ID  | DMA Channel | Sensitivity |
|-----------------------------------|-----|-------------|-------------|
| USB0 DMA Status/Transfer Complete | 123 |             | LEVEL       |

## ADSP-BF60x USB Trigger List

**Table 22-4:** Masters ADSP-BF60x USB Trigger List

| Description                       | ID | Sensitivity |
|-----------------------------------|----|-------------|
| USB0 DMA Status/Transfer Complete | 69 | LEVEL       |
|                                   |    |             |

**Table 22-5:** Slaves ADSP-BF60x USB Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

## USB Block Diagram

The USB block diagram shows the functional blocks within the USB. For more information about the blocks, see the [USB Functional Description](#).

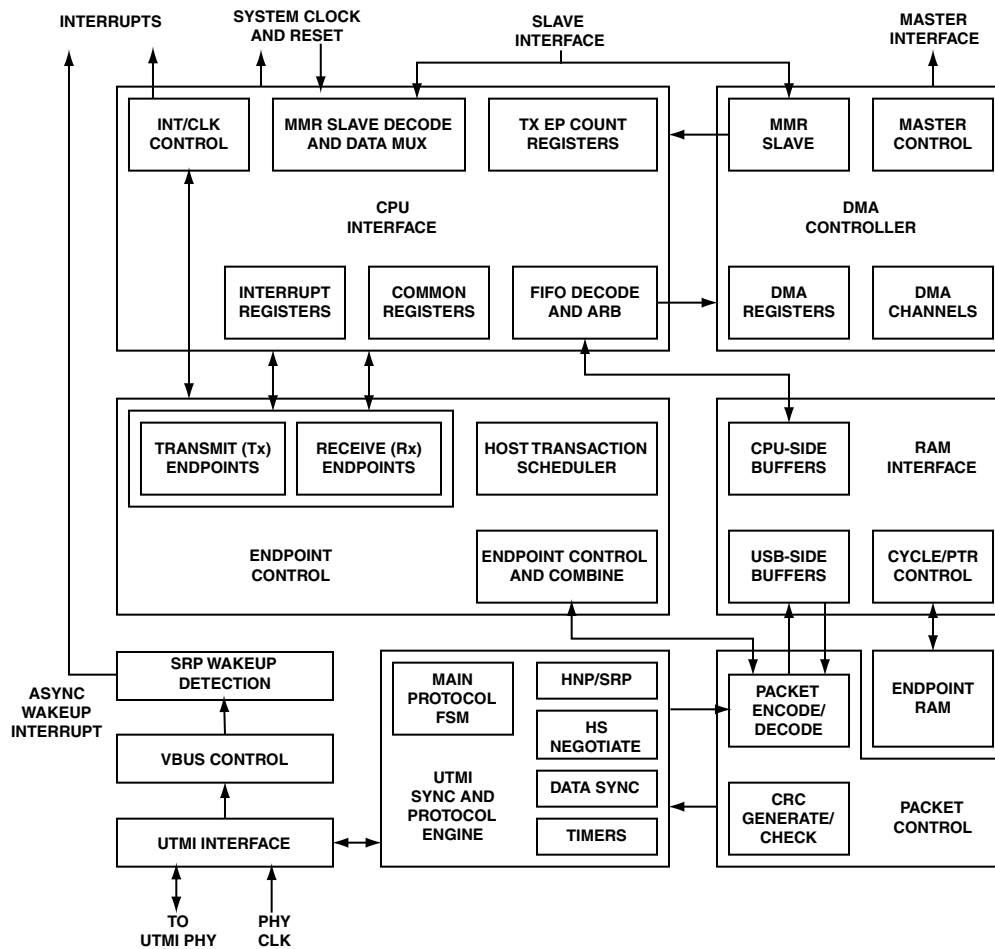


**Figure 22-1:** USB OTG Controller Block Diagram



## USB Definitions

A list of common USB terms and their definitions as used in this specification and with respect to the USB controller follows:

### 'A' Device

The USB device with a mini-A plug inserted into its receptacle. The 'A' device always supplies power to VBUS.

### 'B' Device

The USB device with a standard-B or mini-B plug inserted into its receptacle. The B device starts a session as the peripheral.

### Bi-directional endpoint

An endpoint that can concurrently support receive and transfer packets.

### Control endpoint

An endpoint that is solely used for transfer of USB control packets for setup and configuration. In all USB devices, the control endpoint refers to the bi-directional endpoint 0.

### Dual role device

A USB device that can operate either as the USB host in an OTG session or as a traditional USB peripheral.

### Endpoint

A single physical communication channel for USB, implemented as a FIFO and control logic for that endpoint. Each endpoint has an associated USB transfer type, maximum packet size, bandwidth requirement, endpoint number, and (often) a fixed transfer direction.

### Frame

A regular, fixed 1ms time slot that can contain several transactions. The transfer type determines what transactions are permitted for a given endpoint.

### HNP

Host negotiation protocol. Part of the USB OTG Supplement that allows the host function to be transferred between two connected dual role devices.

### Packet

The lowest level of data exchange on USB. The size is determined by the transfer type and buffer size of the USB peripheral.

### PHY

The PHY is a transceiver circuit that implements the physical layer of USB. For full speed USB OTG this

includes line drivers and receivers, pull-up/pull-down resistors as well as device ID and VBUS level detection.

### **Session**

A period during which USB transfers take place within an OTG connection. This can be initiated by the 'A' device (by driving VBUS) or 'B' device (by initiating SRP). VBUS is powered during a session.

### **SRP**

Session request protocol. Part of the USB OTG Supplement that allows a 'B' device to turn on VBUS and initiate a USB session.

### **Transaction**

Collection of one or more packets in sequence

### **Transfer**

Collection of one or more transfers in sequence

### **Unidirectional endpoint**

Endpoint with its direction fixed in a single direction (for example, it can only receive packets from the USB) in both host and peripheral modes.

## **USB References**

The following references provide further information regarding the USB.

- *On-The-Go Supplement to the USB 2.0 Specification*, Rev 1.0a, June 24, 2003, USB-IF
- *Universal Serial Bus Specification 2.0*

## **USB Operating Modes**

The USB OTG interface may operate in peripheral mode or host mode.

When the USB controller is operating in peripheral mode, the controller may be attached to a conventional host (such as a personal computer) or another OTG device operating in host mode. The second device can be high-speed or full-speed. When linked to another peripheral device, the USB controller can also act as the host, and if the other device is also a dual role controller, the two devices can switch roles as required.

The role taken by the USB controller depends on the way the devices are cabled together. Each USB cable has an 'A' and a 'B' device end. If the 'A' end of the cable is plugged into the device containing the USB controller, the USB controller takes the role of the host device and goes into host mode (in this case the

HOST\_MODE bit is set to 1). If the 'B' of the cable is plugged in, the USB controller goes instead into peripheral mode (and the HOST\_MODE bit remains at 0).

When both devices contain dual role controllers, signaling may be used to switch the roles of the two devices, without switching the cable connecting the two devices. The conditions under which the USB controller may switch between peripheral and host mode are detailed in [Host Negotiation/Configuration](#).

**NOTE:** The USB controller's Multi-Point capability is associated with a range of registers recording the allocation of device functions to individual USB controller's endpoints and device function characteristics such as endpoint number, operating speed and transaction type on an endpoint-by-endpoint basis. Although principally associated with the use of the USB controller as the host to a number of devices, these registers also need to be set when the core is used as the host for a single target device.

To enable the USB:

1. Configure the USB PLL multiplier settings in the USB PLL Control register. Check the processor data sheet for the input clock frequency requirements.
2. Enable the USB PHY by setting the enable bit (bit 7) in the PHY control register.
3. Poll the PLL\_STABLE bit in the USB PLL control register to ensure that the USB PLL has locked to the new frequency.

## USB Peripheral Mode

USB OTG interface operations for the peripheral mode differ from host mode in a number of ways. The following sections describe peripheral mode operations.

### Endpoint Setup

In peripheral mode, there are a few endpoint-specific configuration bits that are used when setting up an endpoint for transfer for all types of peripheral transfer. They determine how the processor core interacts with the endpoint FIFO.

One key parameter required before transfer can occur through an endpoint is the maximum USB packet size that the endpoint can support. This value is set by the software and depends on a variety of system constraints. These include the size of hardware FIFO available and system latencies as well as the USB transfer type and class being used. The USB\_TXMAXPn or USB\_RXMAXPn defines the maximum amount of data that can be transferred to the selected endpoint in a single frame, and the value must match the programmed maximum individual packet size (MaxPktSize) of the standard endpoint descriptor for the endpoint.

For TX endpoints, the maximum packet size is programmed using the USB\_RXMAXPn. For RX endpoints, the USB\_TXMAXPn register is used. The maximum packet size must not exceed the actual hardware endpoint FIFO size. The sizes of the transmit/receive FIFOs, as well as, single or double buffered mode for Endpoints 1 to 11 are determined by the settings in the corresponding FIFO\_SIZE register. Because the USB

controller uses a 32-bit interface, the value chosen for MaxPktSize should be an even number, as this selection simplifies transferring data between FIFOs and processor core.

Additional setup parameters are configured using the `USB_RXCSR` or `USB_TXCSR` register (depending on whether the endpoint in question is RX or TX). The `DMA_ENA` bit in this register is used to enable the assertion of the appropriate DMA request whenever the endpoint is able to receive or transmit another packet. The `AUTOCLEAR_R` and `AUTOSET_R/T` bits can be used to automatically set the FIFO ready triggers (`RXPKTRDY` and `TXPKTRDY`) whenever a packet is transferred to streamline DMA operation for transfers that span multiple packets. Note, however, that `AUTOCLEAR` and `AUTOSET` cannot be used with high-bandwidth endpoints. Refer to the Register Descriptions section for more details on the endpoint control and status registers.

## IN Transactions as a Peripheral

When the USB controller is operating in peripheral mode, data for IN transactions is handled through the TX FIFOs. The maximum size of data packet that may be placed in a TX endpoint's FIFO for transmission is programmable and (where applicable) is determined by the value written to the `USB_TXMAXPn` register for that endpoint (maximum payload multiplied by the number of transactions per micro-frame).

The maximum packet size set for any endpoint must not exceed the FIFO size. (See [USB FIFO Configuration](#).)

**ATTENTION:** The `USB_TXMAXPn` register should not be written-to while there is data in the FIFO, as unexpected results may occur.

For *single packet buffering*, as each packet to be sent is loaded into the TX FIFO, the `TXPKTRDY` bit in `USB_TXCSR` needs to be set. If the `AUTOSET_T` bit in `USB_TXCSR` is set, the `TXPKTRDY` bit is automatically set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, and where Auto Set may not be used (high-bandwidth Isochronous/Interrupt transactions) `TXPKTRDY` always has to be set manually (for example, set by the processor core).

When the `TXPKTRDY` bit is set, either manually or automatically, the `FIFO_NOT_EMPTY_T` bit in `USB_TXCSR` is also set and the packet is ready to be sent. When the packet is successfully sent, both `TXPKTRDY` and `FIFO_NOT_EMPTY_T` are cleared and the appropriate TX endpoint interrupt is generated (if enabled). The next packet can then be loaded into the FIFO.

For *double packet buffering*, as each packet to be sent is loaded into the TX FIFO, the `TXPKTRDY` bit in `USB_TXCSR` needs to be set. If the `AUTOSET_T` bit in `USB_TXCSR` is set, the `TXPKTRDY` bit automatically is set when a maximum-sized packet is loaded into the FIFO. For packet sizes less than the maximum, `TXPKTRDY` always has to be set manually (for example, set by the processor core). When the `TXPKTRDY` bit is set, either manually or automatically, the `FIFO_NOT_EMPTY_T` bit in `USB_TXCSR` also is set. The `TXPKTRDY` is then immediately cleared (and an interrupt generated, if enabled). A second packet can now be loaded into the TX FIFO and `TXPKTRDY` set again (either manually or automatically if the packet is the maximum size). Both packets are now ready to be sent.

When the first packet is successfully sent, `TXPKTRDY` is cleared and the appropriate TX endpoint interrupt is generated (if enabled) to signal that another packet can now be loaded into the TX FIFO. The state of the

`FIFO_NOT_EMPTY_T` bit at this point indicates how many packets may be loaded. If the `FIFO_NOT_EMPTY_T` bit is set then there is another packet in the FIFO and only one more packet can be loaded. If the `FIFO_NOT_EMPTY_T` bit is cleared then there are no packets in the FIFO and two more packets can be loaded.

## OUT Transactions as a Peripheral

When the USB controller is operating in peripheral mode, data for OUT transactions are handled through the USB controller's RX FIFOs.

The maximum amount of data received by an RX endpoint in any frame or micro-frame (in high-speed mode) is programmable and is determined by the value written to the `USB_RXMAXPn` register for that endpoint. This is the maximum payload multiplied by the number of transactions per micro-frame (where applicable). The maximum packet size must not exceed the FIFO size.

If the size of the RX endpoint FIFO is less than twice the maximum packet size for this endpoint (as set in the `USB_RXMAXPn` register), only one data packet can be buffered in the FIFO and single buffering is selected. When a packet is received and placed in the RX FIFO, the `RXPKTRDY` bit and the `FIFO_FULL` bit in `USB_RXCSR` are set and the appropriate RX endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO. After the packet is unloaded, the `RXPKTRDY` bit needs to be cleared in order to allow further packets to be received. If the `AUTOCLEAR_R` bit in `USB_RXCSR` is set and a maximum-sized packet is unloaded from the FIFO, the `RXPKTRDY` bit is cleared automatically. The `FIFO_FULL` bit is also cleared. For packet sizes less than the maximum, `RXPKTRDY` always has to be cleared manually (for example, set by the processor core).

If double packet buffering is enabled, then two data packets can be buffered. When the first packet to be received is loaded into the RX FIFO, the `RXPKTRDY` bit in `USB_RXCSR` is set and the appropriate RX endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO. Note that the `FIFO_FULL` bit in `USB_RXCSR` is not set at this point. This bit is only set if a second packet is received and loaded into the RX FIFO.

After the first packet is unloaded, `RXPKTRDY` needs to be cleared in order to allow further packets to be received. If the `AUTOCLEAR_R` bit in the `USB_RXCSR` register is set and a maximum-sized packet is unloaded from the FIFO, the `RXPKTRDY` bit is cleared automatically. For packet sizes less than the maximum, `RXPKTRDY` always has to be cleared manually (for example, set by the processor core).

If the `FIFO_FULL` bit was set to 1 when `RXPKTRDY` is cleared, the USB controller first clears the `FIFO_FULL` bit. The controller then sets `RXPKTRDY` again to indicate that there is another packet waiting in the FIFO to be unloaded.

## High-Bandwidth Isochronous/Interrupt Transactions

High-Bandwidth Isochronous/Interrupt transactions use much the same protocol as other Isochronous/Interrupt transactions. There are, however, some special features to conducting High-Bandwidth transactions.

- When setting the Maximum Packet size handled by the endpoint in the TxMaxP/RxMaxP register, the maximum number of transactions per microframe also needs to be set via bits D11 and D12 of that register.

This maximum number of transactions (2 or 3) also represents the maximum number of sections in which any single 'High-Bandwidth' packet can be transferred, which in turn sets the maximum size of packet to 2 or 3 times the maximum payload specified for the endpoint in the same register.

**NOTE:** The maximum payload that can be sent in any transaction is 1Kbyte.

- When sending packets, TxPktRdy needs to be set by the application software. Similarly, when unloading packets from the Rx endpoint FIFO, RxPktRdy needs to be cleared by the application software.

The AutoSet and AutoClear functions cannot be used to set and clear these bits in High-Bandwidth transactions.

- The transmission of packets as a number of sections introduces a further type of error – the transmission of Incomplete packets.

For Tx endpoints, the issue principally applies when the is in Peripheral mode and occurs when the fails to receive enough IN tokens from the host to send all the parts of the data packet. It can also apply to High-Bandwidth Interrupt transactions in Host mode where the core does not receive any response from the device to which the packet is being sent. In both cases, the will set the IncompTx bit in the TxCSR register (D7).

For Rx endpoints, the issue occurs when the PIDs of the received parts of the data packet show that one or more parts of the data packet has not been received. When this happens, the sets the IncompRx bit in the RxCSR register (D8). In the main, this bit will only be set when the core is operating in Peripheral mode but it can also be set in Host mode if (and only if) the device with which the is communicating fails to respond in accordance with the USB protocol.

## High Bandwidth Isochronous/Interrupt IN Endpoints

In high-speed mode, transmit endpoints set up for High-Bandwidth Isochronous/Interrupt transactions can transmit up to three USB packets in any micro-frame, with a payload of up to 1024 bytes in each packet, corresponding to a data transfer rate of up to 3072 bytes per micro-frame.

The following figure provides an overview of high-bandwidth IN endpoints in USB.

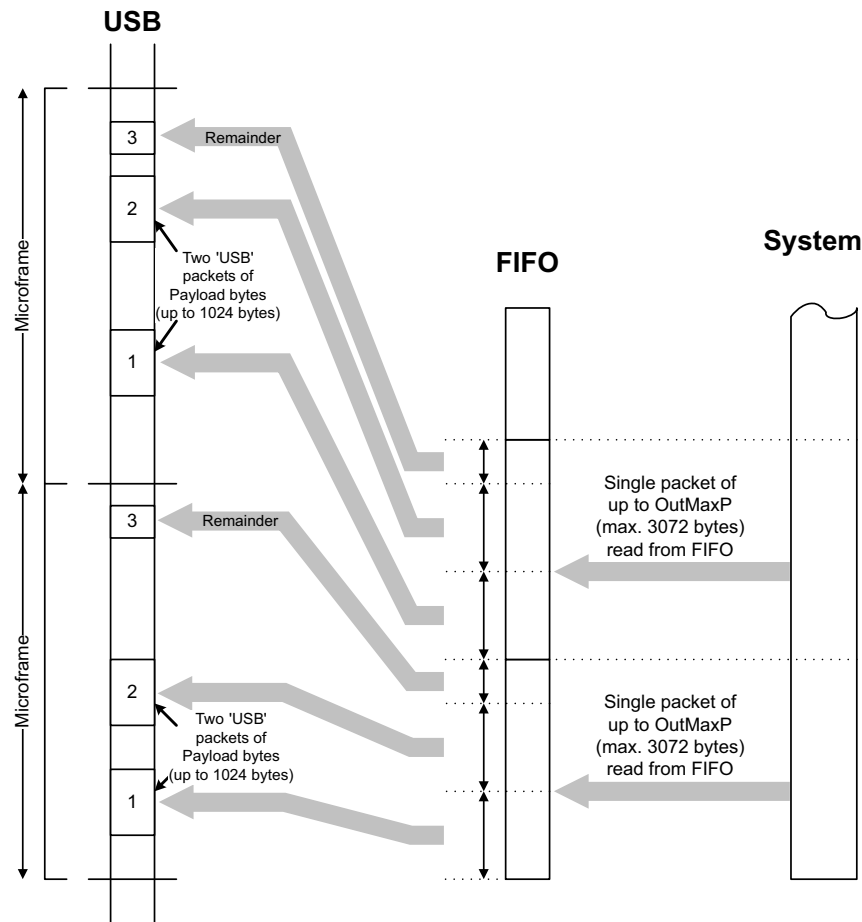


**Figure 22-2:** High Bandwidth IN Endpoints

The USB controller supports these transfers by permitting the loading of data packets with up to 3 times the normal packet size into the associated FIFO in a single transaction. From the point of view of the software in the processor core, the operation is then exactly as described above for single packet buffering or double packet buffering (as appropriate) except that `TXPKTRDY` always needs to be set manually (for example, set by the processor core) as the auto set feature does not operate with high-bandwidth isochronous transfers.

Any data packet loaded into the FIFO that is larger than the maximum payload is automatically split into USB packets of the maximum payload, or smaller, for transmission over the USB. The number of USB packets transmitted per micro-frame and the maximum payload in each packet is defined through the `USB_TXMAXPn` register. Bits 10-0 of the `USB_TXMAXPn` register determine the maximum payload in any USB packet while bits 12,11 determine the maximum number of such packets that can be sent in one micro-frame (2 or 3). Together, these set the maximum size of packet that can be loaded into the FIFO.

At least one USB packet always is sent. The number of further USB packets sent in the same micro-frame depends on the amount of data loaded into the FIFO. The `TXPKTRDY` bit is cleared and an interrupt is generated only when all the packets have been sent. Each USB packet is sent in response to an IN token. If, at the end of a micro-frame, the USB controller has not received enough IN tokens to send all the USB packets (for example, because one of the IN tokens received was corrupted), the remaining data is flushed

from the FIFO. The `TXPKTRDY` bit is cleared and the `INCOMPTX_T` bit in the `USB_TXCSR` register is set to indicate that not all of the data loaded into the FIFO was sent.

## High Bandwidth Isochronous/Interrupt OUT Endpoints

In high-speed mode, isochronous receive endpoints can receive up to three USB packets in any micro-frame, with a payload of up to 1024 bytes in each packet, corresponding to a data transfer rate of up to 3072 bytes per micro-frame. High-Bandwidth Interrupt transactions can similarly be received in Host mode, but note there is no support for high bandwidth Interrupt transactions in Peripheral mode.

The following figure shows an overview of high-bandwidth OUT endpoints.

The USB controller supports this rate by automatically combining all the USB packets received during a micro-frame into a single packet of up to 3 normal packets in size within the Rx FIFO. From the point of view of the software in the processor core, the operation is then exactly as described above for single packet buffering or double packet buffering (as appropriate), except the `RXPKTRDY` always needs to be cleared manually (for example, cleared by the processor core) as `AUTOCLEAR_R` does not operate with high-bandwidth isochronous transfers.

The maximum number of USB packets that may be received in any micro-frame and the maximum payload of these packets are defined through the `USB_RXMAXPn` register. Bits 10-0 of this register determine the maximum payload in any USB packet while bits 12,11 determine the maximum number of these packets that may be received in a micro-frame (2 or 3).



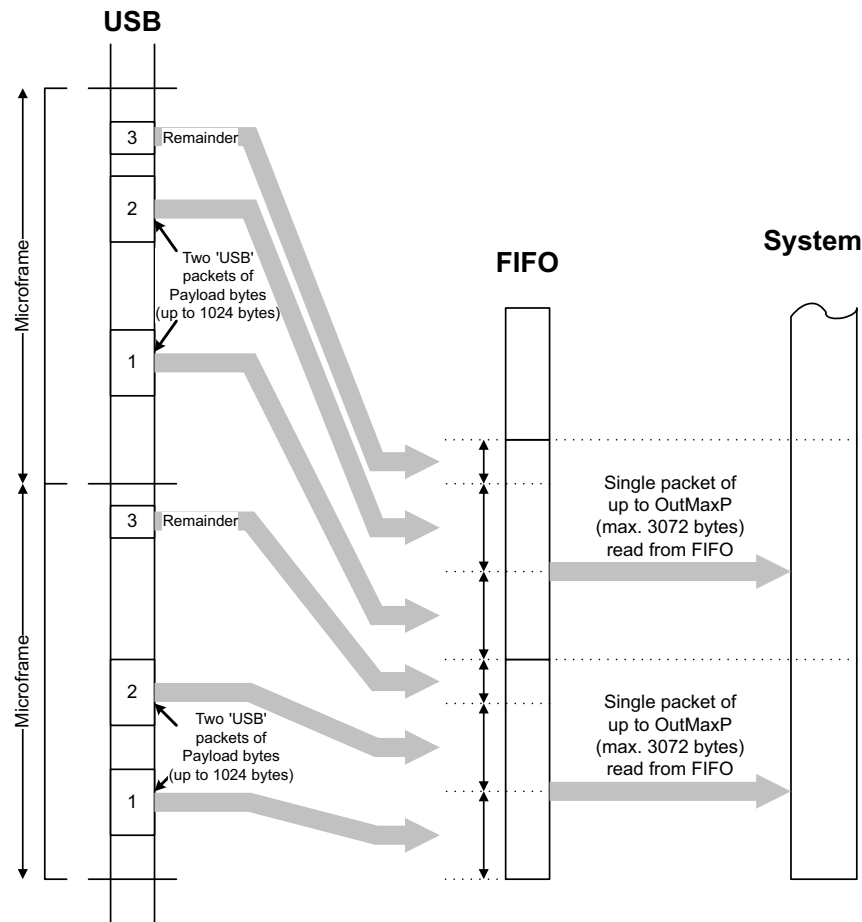


**Figure 22-3:** High Bandwidth OUT Endpoints

The number of USB packets sent in any micro-frame depends on the amount of data to be transferred, and is indicated through the PIDs used for the individual packets. If the indicated number of packets have not been received by the end of a micro-frame, the `INCOMPRX_R` bit in the `USB_RXCSR` register is set to indicate that the data in the FIFO is incomplete. Even so, an interrupt is still be generated to allow the data that has been received to be read from the FIFO.

## Peripheral Transfer Work Flows

The USB transfer types (control, bulk, isochronous and interrupt transfers) each have significantly different system requirements as well as individual USB transfer-specific features. This dictates that they are each dealt with slightly differently in software. For these reasons, there is no uniform way of doing transfers across all transfer types on the USB controller.

The following section provides some guideline peripheral mode transfer flows for each of the transfer types, in both IN (TX) and OUT (RX) directions. In the case of bulk endpoints, the optimal transfer flow differs depending on whether the final size of the transfer is known or unknown. Whether the transfer size is known or not depends on the USB driver class being used. Some define the complete transfer size, and

others operate on a packet-by-packet basis using a short packet (a packet of less than `USB_TXMAXPn` or less than `USB_RXMAXPn`) to denote the end of a transfer.

Each of the work flows use the following common method.

1. Configure the endpoint control and status registers and the `USB_TXMAXPn` or `USB_RXMAXPn` value.
2. Configure the appropriate data transfer mechanism (DMA or interrupt setup).
3. Data transfer phase

The work flows do not describe the USB controller's actions immediately preceding the endpoint setup (for example, the reception of an IN/OUT token from the host, token validity checking, or NAK generation, among others). Note also that there is currently no error-handling contained in the work flows (for example, checking `FIFO_FULL` bit before writing data).

The terms packets, frames and transfers are used in the proceeding sections with their strict USB definitions. (See the [USB Definitions](#) for these definitions.)

## Control Transactions as a Peripheral

Endpoint 0 is the main control endpoint of the USB controller. As such, the routines required to service Endpoint 0 are more complicated than those required to service other endpoints.

The software is required to handle all the Standard Device Requests that may be sent or received through Endpoint 0. These are described in *Universal Serial Bus Specification*, Revision 2.0, Chapter 9. The protocol for these device requests involves different numbers and types of transactions per transfer. To accommodate this, the processor needs to take a state machine approach to command decoding and handling.

The Standard Device Requests received by a USB peripheral can be divided into three categories: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

This section looks at the sequence of actions that the software must perform to process these different types of device request.

## Write Requests

Write requests involve an additional packet (or packets) of data being sent from the host after the 8-byte command. An example of a 'Write' Standard Device Request is: `SET_DESCRIPTOR`.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The `RxPktRdy` bit will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded.

As with a zero data request, the `USB_CSR0` register should then be written to set the `ServicedRxPktRdy` bit (indicating that the command is read from the FIFO) but in this case the `DataEnd` bit should not be set (indicating that more data is expected).

When a second Endpoint 0 interrupt is received, the `USB_CSR0` register is read to check the endpoint status. The `RxPktRdy` bit is set to indicate that a data packet is received. The `USB_COUNT0` register should then be read to determine the size of this data packet. The data packet can then be read from the Endpoint 0 FIFO.

If the length of the data associated with the request (indicated by the `wLength` field in the command) is greater than the maximum packet size for Endpoint 0, further data packets will be sent. In this case, `USB_CSR0` is written to set the `ServicedRxPktRdy` bit, but the `DataEnd` bit should not be set.

When all the expected data packets have been received, the `USB_CSR0` register is written to set the `ServicedRxPktRdy` bit and to set the `DataEnd` bit (indicating that no more data is expected).

When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software-the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the `USB_CSR0` register should be written to set the `ServicedRxPktRdy` bit and to set the `SendStall` bit. When the host sends more data, the USB controller will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the `SentStall` bit will be set.

If the host sends more data after the `DataEnd` has been set, then the USB controller will send a STALL. An Endpoint 0 interrupt will be generated and the `SentStall` bit will be set.

## Read Requests

Read requests have a packet (or packets) of data sent from the function to the host after the 8-byte command. Examples of Standard Device Requests for Read are: `GET_CONFIGURATION`, `GET_INTERFACE`, `GET_DESCRIPTOR`, `GET_STATUS`, `SYNCH_FRAME`.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The `RxPktRdy` bit in `USB_CSR0` will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO and decoded. The `USB_CSR0` register should then be written to set the `ServicedRxPktRdy` bit (indicating that the command has read from the FIFO).

The data to be sent to the host should then be written to the Endpoint 0 FIFO. If the data to be sent is greater than the maximum packet size for Endpoint 0, only the maximum packet size should be written to the FIFO. The `USB_CSR0` register should then be written to set the `TxPktRdy` bit (indicating that there is a packet in the FIFO to be sent). When the packet has been sent to the host, another Endpoint 0 interrupt will be generated and the next data packet can be written to the FIFO.

When the last data packet has been written to the FIFO, the `USB_CSR0` register should be written to set the `TxPktRdy` bit and to set the `DataEnd` bit (indicating that there is no more data after this packet).

When the host moves to the status stage of the request, another Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software-the interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it has been decoded, the `USB_CSR0` register should be written to set the `ServicedRxPktRdy` bit and to set the

SendStall bit. When the host requests data, the USB controller will send a STALL to tell the host that the request was not executed. An Endpoint 0 interrupt will be generated and the SentStall bit will be set.

If the host requests more data after DataEnd has been set, then the USB controller will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit will be set.

## Zero Data Requests

Zero data requests have all their information included in the 8-byte command and require no additional data to be transferred.

Examples of zero data Standard Device Requests are: SET\_FEATURE, CLEAR\_FEATURE, SET\_ADDRESS, SET\_CONFIGURATION, SET\_INTERFACE.

The sequence of events will begin, as with all requests, when the software receives an Endpoint 0 interrupt. The RxPktRdy bit will also have been set. The 8-byte command should then be read from the Endpoint 0 FIFO, decoded and the appropriate action taken. For example if the command is SET\_ADDRESS, the 7-bit address value contained in the command is written to the USB\_FADDR register.

The USB\_CSR0 register should then be written to set the ServicedRxPktRdy bit (indicating that the command is read from the FIFO) and to set the DataEnd bit (indicating that no further data is expected for this request).

When the host moves to the status stage of the request, a second Endpoint 0 interrupt will be generated to indicate that the request has completed. No further action is required from the software-the second interrupt is just a confirmation that the request completed successfully.

If the command is an unrecognized command, or for some other reason cannot be executed, then when it is decoded, the USB\_CSR0 register is written to set the ServicedRxPktRdy bit and to set the SendStall bit. When the host moves to the status stage of the request, the USB controller will send a STALL to tell the host that the request was not executed. A second Endpoint 0 interrupt will be generated and the SentStall bit will be set.

If the host sends more data after the DataEnd bit is set, then the USB controller will send a STALL. An Endpoint 0 interrupt will be generated and the SentStall bit will be set.

## ENDPOINT 0 States

When the USB is operating as a peripheral, the Endpoint 0 control needs three modes (IDLE, TX and RX) corresponding to the different phases of the control transfer and the states Endpoint 0 enters for the different phases of the transfer. (See [Endpoint 0 Service Routine as Peripheral](#).)

The default mode on power-up or reset should be IDLE. The RxPktRdy bit becoming set when Endpoint 0 is in IDLE state indicates a new device request. Once the device request is unloaded from the FIFO, the USB decodes the descriptor to find whether there is a data phase and, if so, the direction of the data phase of the control transfer (in order to set the FIFO direction).

Depending on the direction of the data phase, Endpoint 0 goes into either TX state or RX state. If there is no data phase, Endpoint 0 remains in IDLE state to accept the next device request.

The processor needs to take different actions at the different phases of the possible transfers (for example, "Loading the FIFO", "Setting TxPktRdy") are indicated in the [Endpoint 0 Control States](#) figure. Note that the USB changes the FIFO direction depending on the direction of the data phase, independently of the processor.

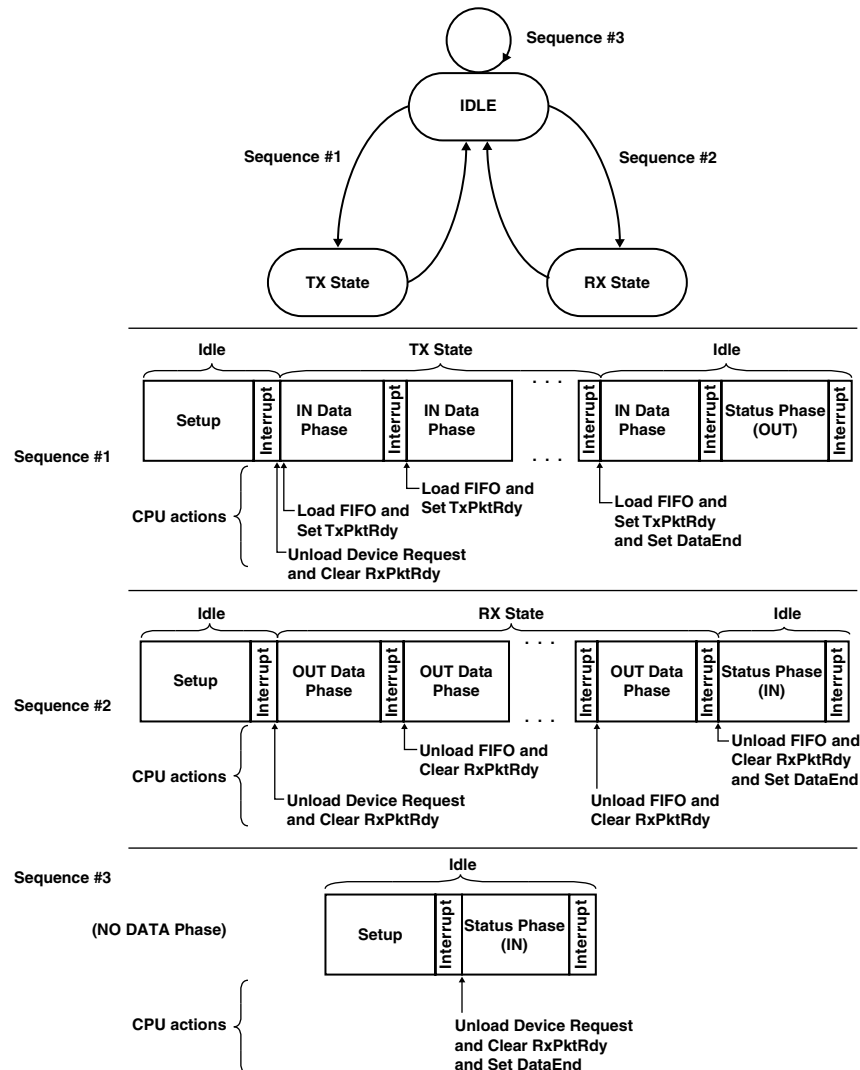


**Figure 22-4:** Endpoint 0 Control States

## Endpoint 0 Service Routine as Peripheral

An Endpoint 0 interrupt is generated:

- When the USB controller sets the RxPktRdy bit after a valid token has been received and data has been written to the FIFO.
- When the USB controller clears the TxPktRdy bit after the data packet in the FIFO has been successfully transmitted to the host.

- When the USB controller sets the `SentStall` bit after a control transaction is ended due to a protocol violation.
- When the USB controller sets the `SetupEnd` bit because a control transfer has ended before `DataEnd` is set.

The bits mentioned above, are in the `USB_CSR0` register.

Whenever the Endpoint 0 service routine is entered, the firmware must first check whether the current control transfer has been ended due to either a STALL condition or a premature end-of-control transfer. If the control transfer ends due to a STALL condition, the `SentStall` bit would be set. If the control transfer ends due to a premature end-of-control transfer, the `SetupEnd` bit would be set. In either case, the firmware should abort processing the current control transfer and set the state to IDLE.

Once the firmware has determined that the interrupt was not generated by an illegal bus state, the next action depends on the Endpoint state.

If Endpoint 0 is in IDLE state, the only valid reason an interrupt can be generated is as a result of the core receiving data from the USB bus. The service routine must check for this by testing the `RxPktRdy` bit. If this bit is set, then the core has received a SETUP packet. This must be unloaded from the FIFO and decoded to determine the action the core must take. Depending on the command contained within the SETUP packet, Endpoint 0 will enter one of three states:

- If the command is a single packet transaction (`SET_ADDRESS`, `SET_INTERFACE` etc.) without a data phase, the endpoint will remain in IDLE state.
- If the command has an OUT data phase (`SET_DESCRIPTOR` etc.), the endpoint will enter RX state.
- If the command has an IN data phase (`GET_DESCRIPTOR` etc.), the endpoint will enter TX state.

If the endpoint is in TX state, the interrupt indicates that the core has received an IN token and data from the FIFO has been sent. The firmware must respond to this either by placing more data in the FIFO if the host is still expecting more data<sup>1</sup> or by setting the `DataEnd` bit to indicate that the data phase is complete. Once the data phase of the transaction has been completed, Endpoint 0 should be returned to IDLE state to await the next control transaction.

If the endpoint is in RX state, the interrupt indicates that a data packet has been received. The firmware must respond by unloading the received data from the FIFO. The firmware must then determine whether it has received all of the expected data. If it has, the firmware should set the `DataEnd` bit and return Endpoint 0 to IDLE state. If more data is expected, the firmware should set the `ServicedRxPktRdy` bit to indicate that it has read the data in the FIFO and leave the endpoint in RX state.

---

1. Command transactions all include a field that indicates the amount of data the host expects to receive or is going to send.

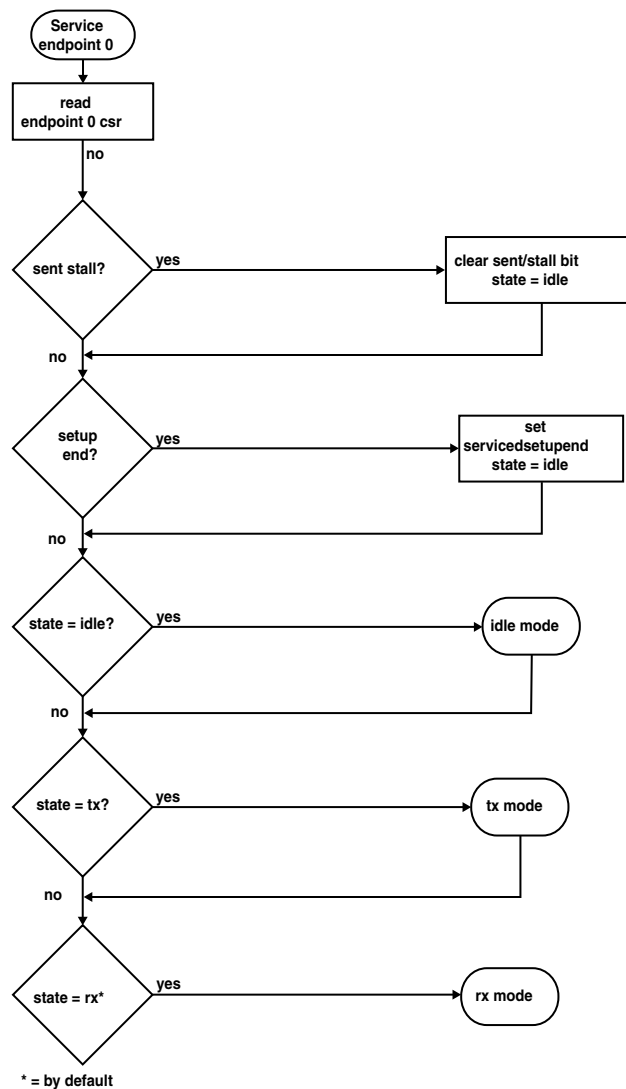


**Figure 22-5:** Endpoint 0 Service Routine

## Idle Mode

The Endpoint 0 control must select the IDLE mode at power-on or reset. The Endpoint 0 control should return to this mode when the RX and TX modes are terminated.

This is also the mode in which the SETUP phase of control transfer is handled. (See the [Endpoint 0 Idle Mode \(Setup Phase\)](#) figure.)

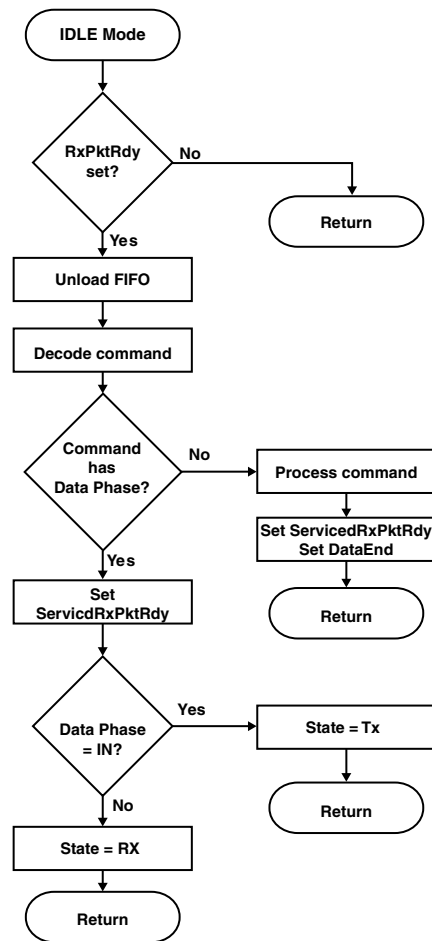


**Figure 22-6:** Endpoint 0 Idle Mode (Setup Phase)

## TX Mode

When the endpoint is in TX state, all arriving IN tokens need to be treated as part of a data phase until the required amount of data has been sent to the host. If either a SETUP or an OUT token is received while the endpoint is in the TX state, a `SetupEnd` condition would occur since the core expects only IN tokens.

Three events can cause the TX mode to terminate before the expected amount of data has been sent:

- The host sends an invalid token causing a `SetupEnd` bit set.
- The firmware sends a packet containing less than the maximum packet size for Endpoint 0.
- The firmware sends an empty data packet.

Until the transaction is terminated, when the firmware receives an interrupt which indicates that a packet has been sent from the FIFO, it simply loads the FIFO. An interrupt is generated when `TxPktRdy` is cleared.

When the firmware forces the termination of a transfer (by sending a short or empty data packet), it should set the `DataEnd` bit to indicate to the core that the data phase is complete and that the core should receive an acknowledge packet next.



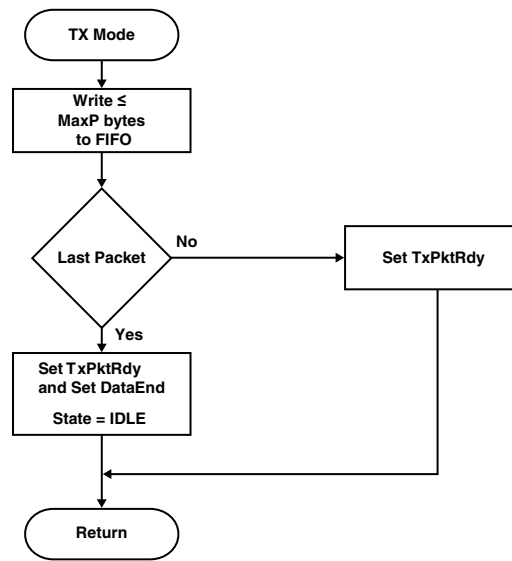


**Figure 22-7:** Endpoint 0 TX Mode

## RX Mode

In RX mode, all arriving data should be treated as part of a data phase until the expected amount of data has been received. If either a SETUP or an IN token is received while the endpoint is in RX state, a SetupEnd condition would occur since the core expects only OUT tokens.

Three events can cause the RX mode to terminate before the expected amount of data has been received:

- The host sends an invalid token causing a SetupEnd bit set.
- The host sends a packet which contains less than the maximum packet size for Endpoint 0.
- The host sends an empty data packet.

Until the transaction is terminated, when the firmware receives an interrupt which indicates that new data has arrived (RxPktRdy bit set), it simply needs to unload the FIFO and clear RxPktRdy by setting the ServicedRxPktRdy bit.

When the firmware detects the termination of a transfer (by receiving either the expected amount of data or an empty data packet), it should set the DataEnd bit to indicate to the core that the data phase is complete and that the core should receive an acknowledge packet next.

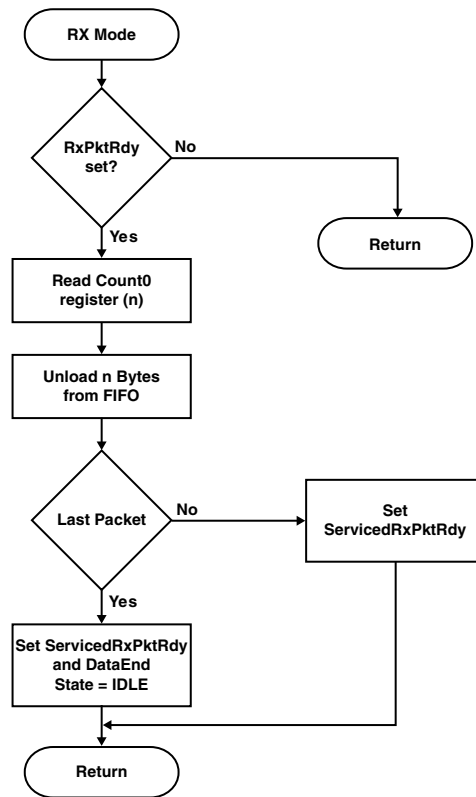


**Figure 22-8:** Endpoint 0 RX Mode

### Peripheral Mode, Bulk IN, Transfer Size Known

*PREREQUISITE:*

For this process, the maximum individual packet size (*MaxPktSize*) in bytes and the complete transfer size (*TxferSize*) in bytes, must be known.

1. Load *MaxPktSize* into USB\_TXMAXPn.
2. Set DMA\_ENA = 1, AUTOSET\_T = 1, ISO\_T = 0, FRCDATATOG = 0 in USB\_TXCSR.
3. Load *TxferSize* into USB\_DMA\_CNTn.
4. Configure the DMA controller to write the data into the corresponding TX FIFO address.
5. On each USB\_DMAxINT transition, the DMA controller writes a new packet into the FIFO. TXPKTRDY is automatically set when each new packet is written.
6. Step 5 is repeated for each full packet of the transfer.
7. Even if the final packet is a short packet, the packet automatically is detected by the USB controller because TXPKTRDY is set.

## Peripheral Mode, Bulk IN, Transfer Size Unknown

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes is assumed to be an even number of bytes.

1. Load *MaxPktSize* into USB\_TXMAXPn.
2. Set DMA\_ENA = 1, AUTOSET\_T = 1, ISO\_T = 0, FRCDATATOG = 0 in USB\_TXCSR.
3. Configure the DMA controller to write *MaxPktSize*/2 half words into the corresponding TX FIFO address on each USB\_DMAxINT.
4. Set up an ISR, sensitive to the DMA work-block-complete interrupt, that writes a remaining short packet into the TX FIFO using processor core DMA. Then set TXPKTRDY or simply send a zero-length packet by toggling TXPKTRDY.
5. On each USB\_DMAxINT transition, the DMA controller writes a new packet into the FIFO. TXPKTRDY automatically is set when each new packet is written.
6. Step 5 is repeated for each full packet of the transfer.
7. The final short/zero-length packet is managed by the ISR from step 4.

## Peripheral Mode, ISO IN, Small MaxPktSize

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes is less than 128 bytes and is an even number of bytes. Double buffering is assumed to be enabled, and the auto set feature unused (because packets are often less than *MaxPktSize*).

1. Load *MaxPktSize* into USB\_TXMAXPn.
2. Set ISO\_T = 1 in USB\_TXCSR.
3. Preload the first two packets into the endpoint TX FIFO and set TXPKTRDY.
4. Set up an ISR, sensitive to the SOF\_B interrupt, which writes a new packet into the TX FIFO and sets TXPKTRDY.
5. Set SOF\_B = 1 in USB\_INTRUSBE to generate an interrupt on each start-of-frame.
6. Step 5 is repeated for each ISO packet.

## Peripheral Mode, ISO IN, Large MaxPktSize

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes is greater than 128 bytes and is an even number of bytes. Double buffering is assumed to be enabled, and the auto set feature unused (because packets are often less than *MaxPktSize*).

1. Load *MaxPktSize* into USB\_TXMAXPn.
2. Set ISO\_T = 1 in USB\_TXCSR.
3. Set ISO\_UPDATE = 1 in USB\_POWER to prevent initial packet loaded into the FIFO from being transmitted on USB until the next 1ms frame.
4. Load the total number of bytes for the first two packets into USB\_DMA\_CNTn.
5. Configure the DMA controller to pre-load the two packets into the corresponding TX FIFO address. Set TXPKTRDY.
6. Set up an ISR, sensitive to the SOF\_B interrupt, which writes a new packet into the TX FIFO by configuring the DMA controller to load the packet.
7. Set SOF\_B = 1 in USB\_INTRUSBE to generate an interrupt on each start-of-frame.
8. Step 7 is repeated for each ISO packet.

## Peripheral Mode, Bulk OUT, Transfer Size Known

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes and the complete transfer size (*TxferSize*) in bytes must be known.

1. Load *MaxPktSize* into USB\_RXMAXPn.
2. Set DMA\_ENA = 1, AUTOCLEAR\_R = 1, ISO\_R = 0, FRCDATATOG = 0, DMAREQMODE\_R = 0 in USB\_RXCSR.
3. Configure the DMA controller to read the full *TxferSize*/2 half words from the corresponding RX FIFO address.
4. On each USB\_DMAxINT transition, the DMA controller reads another packet from the FIFO. RXPKTRDY is automatically cleared by the USB controller when each new packet is read.
5. Step 5 is repeated for each full packet of the transfer.
6. If *TxferSize* is not an exact multiple of *MaxPktSize*, the final USB\_DMAxINT transition causes the DMA controller to read out only the short packet that remains.

## Peripheral Mode, Bulk OUT, Transfer Size Unknown

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes must be known.

1. Load *MaxPktSize* into USB\_RXMAXPn.
2. Set DMA\_ENA = 1, AUTOCLEAR\_R = 1, ISO\_R = 0, FRCDATATOG = 0, DMAREQMODE\_R = 1 in USB\_RXCSR.
3. Set the appropriate EPx\_RX\_E bit in USB\_INTRRXE.
4. Configure the DMA controller to read *MaxPktSize*/2 half words from the corresponding RX FIFO address on each USB\_DMAxINT transition.
5. Set up an ISR, sensitive to the RX interrupt, which reads USB\_RXCOUNT and then transfers USB\_RXCOUNT bytes (in half words) from the RX FIFO to the processor core. Depending on the number of bytes in the FIFO, this can be performed by configuring the DMA to read the data, or by reading it with the processor core.
6. On each USB\_DMAxINT transition, the DMA controller reads a packet from the FIFO. RXPkTRDY is automatically cleared by the USB controller when each new packet is read.
7. Step 5 is repeated for each full packet of the transfer.
8. If a packet is received that is less than *MaxPktSize*, the RX interrupt goes high, and the ISR from step 5 reads out the remaining short packet.

## Peripheral Mode, ISO OUT, Small MaxPktSize

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes is less than 128 bytes, and double buffering is assumed to be enabled.

1. Load *MaxPktSize* into USB\_RXMAXPn.
2. Set ISO\_R = 1 in USB\_RXCSR.
3. Set up an ISR, sensitive to the SOF\_B interrupt, that reads the FIFO\_FULL bit, reads the USB\_RXCOUNT status register, and finally removes one or two packets (equal to the USB\_RXCOUNT number of bytes) from the FIFO then clears RXPkTRDY.
4. Set SOF\_B = 1 in USB\_INTRUSBE to generate an interrupt on each start-of-frame.
5. Step 4 is repeated for each ISO packet.

## Peripheral Mode, ISO OUT, Large MaxPktSize

### PREREQUISITE:

For this process, the maximum individual packet size (*MaxPktSize*) in bytes is greater than 128 bytes, and double buffering is assumed to be enabled.

1. Load *MaxPktSize* into the USB\_RXMAXPn register.
2. Set the ISO\_R = 1 in the USB\_RXCSR register.
3. Set up an ISR, (sensitive to the SOF\_B interrupt), that reads the FIFO\_FULL bit, reads the USB\_RXCOUNT status register, and finally configures the DMA controller to remove one or two packets (equal to the USB\_RXCOUNT number of bytes) from the FIFO.
4. Set up an ISR, sensitive to the DMA work-block-complete interrupt to clear RXPKTRDY.
5. Set SOF\_B = 1 in the USB\_INTRUSBE register to generate an interrupt on each start-of-frame.
6. Step 5 is repeated for each ISO packet.

## Peripheral Mode Suspend

When no activity has occurred on the USB for 3 ms, the USB controller enters suspend mode. If the suspend interrupt (SUSPEND\_B) is enabled, an interrupt is generated at this time.

When resume signaling is detected, the USB controller exits suspend mode. If the RESUME\_B interrupt is enabled, an interrupt is generated. The processor core can also force the USB controller to exit suspend mode by setting the RESUME\_MODE bit in the USB\_POWER register. This will initiate a remote wakeup. When this bit is set, the USB controller exits suspend mode and drives resume signaling onto the bus. The processor core should clear this bit after 10 ms (a maximum of 15 ms) to end resume signaling.

**NOTE:** No Resume interrupt will be generated when the software initiates remote wakeup.

No RESUME\_B interrupt is generated when suspend mode is exited by the processor core.

## Start-of-frame (SOF) Packets

When the USB controller is operating in peripheral mode, it should receive a start-of-frame packet from the host every millisecond when in full-speed mode, or every 125 microseconds when in high-speed mode.

When the SOF packet is received, the 11-bit frame number contained in the packet is written into the USB\_FRAME register and an output pulse, lasting one USB clock bit period, is generated on SOF\_PULSE (internal USB controller signal). A SOF\_B interrupt is also generated (if enabled in the USB\_INTRUSBE register).

After the USB controller has started to receive SOF packets, the controller expects one every millisecond (or 125 ms when in high-speed mode). If no SOF packet is received after 1.00358 ms (or 125.125 ms), it is assumed that the packet is lost. An SOF\_PULSE (together with a SOF\_B interrupt, if enabled) is still generated though the USB\_FRAME register is not updated. The USB controller continues to generate an SOF\_PULSE every millisecond (or 125 ms) and re-synchronizes these pulses to the received SOF packets when these packets are successfully received again.

## Soft Connect/Soft Disconnect

In peripheral mode, the USB controller can be programmed to switch between normal mode and non-driving mode by setting or clearing the SOFT\_CONN bit of the USB\_POWER register. When this SOFT\_CONN bit is set to 1, the USB controller is placed in its normal mode and the D+/D- lines of the USB bus are enabled. When the SOFT\_CONN bit is zero, the PHY is put into non-driving mode and D+ and D- are three-stated. The USB controller appears to have been disconnected from the USB bus.

After system reset, SOFT\_CONN is cleared to 0. From that point, the USB controller appears disconnected until the software has set SOFT\_CONN to 1. The application software can then choose when to set the PHY to its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB. Once the Soft Conn bit has been set to 1, the software can also simulate a disconnect by clearing this bit to 0.

## Error Handling As a Peripheral

A control transfer may be aborted due to a protocol error on the USB, the host prematurely ending the transfer, or if the function controller software wishes to abort the transfer (for example, because it cannot process the command).

The USB controller will automatically detect protocol errors and send a STALL packet to the host under the following conditions:

1. The host sends more data during the OUT data phase of a write request than was specified in the command. This condition is detected when the host sends an OUT token after the DataEnd bit is set.
2. The host requests more data during the IN data phase of a read request than was specified in the command. This condition is detected when the host sends an IN token after the DataEnd bit in the USB\_CSR0 register is set.
3. The host sends more than *MaxPktSize* data bytes in an OUT data packet.
4. The host sends a non-zero length DATA1 packet during the status phase of a read request.

When the USB controller has sent the STALL packet, it sets the SentStall bit and generates an interrupt. When the software receives an Endpoint 0 interrupt with the SentStall bit set, it should abort the current transfer, clear the SentStall bit, and return to the IDLE state.

If the host prematurely ends a transfer by entering the status phase before all the data for the request is transferred, or by sending a new SETUP packet before completing the current transfer, then the SetupEnd bit will be set and an Endpoint 0 interrupt generated. When the software receives an Endpoint 0 interrupt

with the `SetupEnd` bit set, it should abort the current transfer, set the `ServicedSetupEnd` bit, and return to the IDLE state. If the `RxPktRdy` bit is set, this indicates that the host has sent another SETUP packet and the software should then process this command.

If the software wants to abort the current transfer, because it cannot process the command or has some other internal error, then it should set the `SendStall` bit. The USB controller will then send a STALL packet to the host, set the `SentStall` bit and generate an Endpoint 0 interrupt.

## Stalls Issued to Control Transfers

In peripheral mode, the USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an OUT token (instead of an IN token) after the processor core has unloaded the last OUT packet and set `DATAEND`.
2. The host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an IN token (instead of an OUT token) after the processor core has cleared `TXPKTRDY` and set `DATAEND` in response to the ACK issued by the host to what should have been the last packet.
3. The host sends more than *MaxPktSize* data with an OUT data token.
4. The host sends the wrong PID (packet identifier) for the OUT status phase of a control transfer.
5. The host sends more than a zero length data packet for the OUT status phase.

## Zero Length OUT Data Packets in Control Transfers

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request is transferred (for example, after the processor core has set `DATAEND`). If the host sends a zero-length OUT data packet before the entire length of device request is transferred, this packet signals the premature end of the transfer. In this case, the USB controller automatically flushes any IN token loaded by processor core ready for the data phase from the FIFO and sets `SETUPEND`.

## USB Host Mode

USB OTG interface operations in host mode differ from peripheral mode in a number of ways. The following sections describe host mode operations.



## Transaction Scheduling

When operating as a host, the USB controller maintains a frame counter. If the target function is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame or micro-frame. If the target function is a low-speed device, a K state is transmitted on the bus to act as a *keep-alive* to stop the low-speed device from going into suspend mode.

After the SOF packet is transmitted, the USB controller cycles through all the endpoints looking for active transactions. An active transaction is defined as an RX endpoint for which the REQPKT\_H bit is set or a TX endpoint for which the TXPKTRDY bit is set.

An active isochronous or interrupt transaction will only start if it is found on the first transaction scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. This ensures that only one interrupt or isochronous transaction occurs per endpoint per  $n$  frames/micro frames (or up to three if high-bandwidth support is selected) where  $n$  is the interval set in the USB\_TXINTERVAL or USB\_RXINTERVAL register for that endpoint.

An active bulk transaction is started immediately, provided there is sufficient time left in the frame to complete the transaction before the next SOF packet is due. If the transaction needs to be retried (for example, because a NAK was received or the target function did not respond) then the transaction is not retried until the transaction scheduler has checked all the other endpoints for active transactions first. This check ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The USB controller lets you specify a limit (USB\_TXINTERVAL or USB\_RXINTERVAL registers) to the length of time in which NAKs may be received from a particular target before the endpoint is timed out.

## Endpoint Setup and Data Transfer

When the HOST\_MODE bit is set to 1, the USB controller operates as a host for point-to-point communications with another USB device or, when attached to a hub, for communication with a whole range of devices in a multi-point set-up. High-speed, full-speed and low-speed USB functions are supported, both for point-to-point communication and for operation through a hub. (Where necessary, the core automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub.)

Control, Bulk, Isochronous or Interrupt transactions are supported.

Transfers between the subsystem and endpoint FIFOs in host mode are similar to peripheral mode. With this in mind, see many of the descriptions of processor core to FIFO data transfer in [USB Peripheral Mode](#).

## Control Transaction as a Host

Host control transactions are conducted through Endpoint 0. The software is required to handle all the Standard Device Requests that may be sent or received through Endpoint 0 (as described in *Universal Serial Bus Specification*, Revision 2.0, Chapter 9).

For a USB peripheral, there are three categories of Standard Device Requests to be handled: Zero Data Requests (in which all the information is included in the command), Write Requests (in which the

command will be followed by additional data), and Read Requests (in which the device is required to send data back to the host).

Zero Data Requests comprise a SETUP command followed by an IN status phase.

Write Requests comprise a SETUP command, followed by an OUT data phase followed by an IN status phase.

Read Requests comprise a SETUP command, followed by an IN data phase followed by an OUT status phase.

A timeout may be set to limit the length of time during which the USB controller will retry a transaction that is continually NAKed by the target. This limit can be between 2 and  $2^{15}$  frames/micro frames and is set through the `USB_NAKLIMIT0` register.

The following sections look at the steps in different phases of a control transaction to describe the actions of the core in issuing Standard Device Requests.

## Setup Phase as a Host

The processor core driving the host device performs the following actions for the SETUP phase of a control transaction.

1. Load the eight bytes of the required device request command into the Endpoint 0 FIFO
2. Set `SETUPPKT_H` (bit 3) and `TxPtdRdy` (bit 1) of the `USB_CSR0` register. These bits must be set together.

The USB controller then sends a SETUP token followed by the 8-byte command to Endpoint 0 of the addressed device, retrying as necessary.

3. At the end of the attempt to send the data, the USB controller will generate an Endpoint 0 interrupt (for example, set `IntrTx.D0`). The processor core should then read `USB_CSR0` to establish whether the `STALL_RECEIVED_H` bit, the `ERROR_H` bit or the `NAK_TIMEOUT_H` bit is set.

If `STALL_RECEIVED_H` is set, it indicates that the target did not accept the command (for example, because it is not supported by the target device) and so has issued a STALL response.

If `ERROR_H` is set, it means that the USB controller has tried to send the SETUP packet and the following data packet three times without getting a response.

If `NAK_TIMEOUT_H` is set, it means that the USB controller has received a NAK response to each attempt to send the SETUP packet, for longer than the time set in the `USB_NAKLIMIT0` register. The USB controller can then be directed either to continue trying this transaction (until it times out again) by clearing the `NAK_TIMEOUT_H` bit or to abort the transaction by flushing the FIFO before clearing the `NAK_TIMEOUT_H` bit.

4. If none of `STALL_RECEIVED_H`, `ERROR_H` or `NAK_TIMEOUT_H` is set, the SETUP phase is correctly acknowledged and the processor core should proceed to the following IN data phase, OUT data phase or IN status phase specified for the particular Standard Device Request.

## IN Data Phase as a Host

The processor core driving the host device performs the following actions for the IN data phase of a control transaction.

1. Set REQPKT\_H in USB\_CSR0.
2. Wait while the USB controller sends the IN token and then receives the required data back.
3. When the USB controller generates the Endpoint 0 interrupt (for example, by setting EPO\_TX in the USB\_INTRTX register)-read USB\_CSR0 to establish whether the STALL\_RECEIVED\_H bit, the ERROR\_H bit, the NAK\_TIMEOUT\_H bit or RxPktRdy is set.

If STALL\_RECEIVED\_H is set, it indicates that the target has issued a STALL response.

If ERROR\_H is set, it means that the USB controller has tried to send the required IN token three times without getting a response.

If NAK\_TIMEOUT\_H is set, it means that the USB controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the USB\_NAKLIMIT0 register. The USB controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT\_H bit; or to abort the transaction by clearing REQPKT\_H before clearing the NAK\_TIMEOUT\_H bit.

4. If RxPktRdy is set, the processor core should read the data from the Endpoint 0 FIFO, then clear RxPktRdy.
5. If further data is expected, the processor core should repeat the previous steps.

When all the data is successfully received, the processor core should proceed to the OUT status phase of the control transaction.

## OUT Data as a Host (Control)

The processor core driving the host device performs the following actions for the OUT data phase of a control transaction.

1. Load the data to be sent into the Endpoint 0 FIFO
2. Set the TxPktRdy bit in USB\_CSR0.

The USB controller then proceeds to send an OUT token followed by the data from the FIFO to Endpoint 0 of the addressed device, retrying as necessary.

3. At the end of the attempt to send the data, the USB controller will generate an Endpoint 0 interrupt (for example, set EPO\_TX in USB\_INTRTX register). The processor core should then read USB\_CSR0 to estab-

lish whether the `STALL_RECEIVED_H` bit (D2), the `ERROR_H` bit (D4) or the `NAK_TIMEOUT_H` bit (D7) is set.

If `STALL_RECEIVED_H` is set, it indicates that the target has issued a `STALL` response.

If `ERROR_H` is set, it means that the USB controller has tried to send the `OUT` token and the following data packet three times without getting a response.

If `NAK_TIMEOUT_H` is set, it means that the USB controller has received a `NAK` response to each attempt to send the `OUT` token, for longer than the time set in the `USB_NAKLIMIT0` register. The USB controller can then be directed either to continue trying this transaction (until it times out again) by clearing the `NAK_TIMEOUT_H` bit; or to abort the transaction by flushing the FIFO before clearing the `NAK_TIMEOUT_H` bit.

If none of `STALL_RECEIVED_H`, `ERROR_H` or `USB_NAKLIMIT0` is set, the `OUT` data is correctly acknowledged.

4. If further data needs to be sent, the processor core should repeat the previous steps.

When all the data is successfully sent, the processor core should proceed to the `IN` status phase of the control transaction.

## IN Status Phase as a Host (Following `SETUP` Phase or `OUT` Data Phase)

The processor core driving the host device performs the following actions for the `IN` status phase of a control transaction.

1. Set `STATUSPKT_H_H` and `REQPKT_H` (bits 6 and 5 of `USB_CSR0`, respectively). These bits must be set together.
2. Wait while the USB controller both sends an `IN` token and receives a response from the USB peripheral.
3. When the USB controller generates the Endpoint 0 interrupt (for example, sets `EP0_TX` in `USB_INTRTX` register), read `USB_CSR0` to establish whether the `STALL_RECEIVED_H` bit, the `ERROR_H` bit, the `NAK_TIMEOUT_H` bit or `RxPktRdy` is set.

If `STALL_RECEIVED_H` is set, it indicates that the target could not complete the command and so has issued a `STALL` response.

If `ERROR_H` is set, it means that the USB controller has tried to send the required `IN` token three times without getting a response.

If `NAK_TIMEOUT_H` is set, it means that the USB controller has received a `NAK` response to each attempt to send the `IN` token, for longer than the time set in the `USB_NAKLIMIT0` register. The USB controller can then be directed either to continue trying this transaction (until it times out again) by clearing the `NAK_TIMEOUT_H` bit or to abort the transaction by clearing `REQPKT_H` and `STATUSPKT_H_H` before clearing the `NAK_TIMEOUT_H` bit.

4. If `RxPktRdy` is set, the processor core should simply clear `RxPktRdy`.

## OUT Status Phase as a Host (following IN Data Phase)

The processor core driving the host device performs the following actions for the OUT status phase of a control transaction.

1. Set STATUSPKT\_H and TxPktRdy bits. These bits must be set together.
2. Wait while the USB controller both sends the OUT token and a zero-length DATA1 packet.
3. At the end of the attempt to send the data, the USB controller will generate an Endpoint 0 interrupt. The processor core should then read USB\_CSR0 to establish whether the STALL\_RECEIVED\_H bit, the ERROR\_H bit or the NAK\_TIMEOUT\_H bit is set.

If STALL\_RECEIVED\_H is set, it indicates that the target could not complete the command and so has issued a STALL response.

If ERROR\_H is set, it means that the USB controller has tried to send the STATUS packet and the following data packet three times without getting a response.

If NAK\_TIMEOUT\_H is set, it means that the USB controller has received a NAK response to each attempt to send the IN token, for longer than the time set in the USB\_NAKLIMIT0 register. The USB controller can then be directed either to continue trying this transaction (until it times out again) by clearing the NAK\_TIMEOUT\_H bit or to abort the transaction by flushing the FIFO before clearing the NAK\_TIMEOUT\_H bit.

4. If none of STALL\_RECEIVED\_H, ERROR\_H or NAK\_TIMEOUT\_H is set, the status phase is correctly acknowledged.

## Host IN Transactions

When the USB controller operates as a host, IN transactions are handled like OUT transactions are handled when the USB controller is operating as a peripheral. But the transaction must first be initiated by setting the REQPKT\_H bit in USB\_RXCSR. This bit indicates to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target function.

When the packet is received and placed in the RX FIFO, the RXPkTRDY bit in USB\_RXCSR is set, and the appropriate RX endpoint interrupt is generated (if enabled) to signal that a packet can now be unloaded from the FIFO. When the packet is unloaded, RXPkTRDY is cleared. The AUTOCLEAR\_R bit in the USB\_RXCSR register can be used to have RXPkTRDY automatically cleared when a maximum sized packet is unloaded from the FIFO. There is also an AUTOREQ\_RH bit in USB\_RXCSR that causes the REQPKT\_H bit to be automatically set when the RXPkTRDY bit is cleared. The AUTOCLEAR\_R and AUTOREQ\_RH bits can be used with an external DMA controller to perform complete bulk transfers without processor core intervention.

If the target function responds to a bulk or interrupt IN token with a NAK, the USB controller keeps retrying the transaction until the NAK limit set (in USB\_NAKLIMIT0) is reached. If the target function responds with a STALL, the USB controller does not retry the transaction, but does interrupt the processor core with the RXSTALL\_TH bit in the USB\_RXCSR register set. If the target function does not respond to the IN token within the required time (or there was a CRC or bit-stuff error in the packet), the USB controller

retries the transaction. If after three attempts the target function still has not responded, the USB controller clears the REQPKT\_H bit and interrupts the processor core with the DATAERROR\_R bit in USB\_RXCSR set.

## Host OUT Transactions

When the USB controller operates as a host, OUT transactions are handled in a similar manner to the way IN transactions are handled when the USB controller operates as a peripheral.

The TXPKTRDY bit in the USB\_TXCSR register needs to be set as each packet is loaded into the TX FIFO and the AUTOSET\_T bit in USB\_TXCSR can be used to cause the TXPKTRDY bit to be automatically set when a maximum sized packet is loaded into the FIFO. The AUTOSET\_T bit can be used with an external DMA controller to perform complete bulk transfers without processor core intervention.

If the target function responds to the OUT token with a NAK, the USB controller keeps retrying the transaction until the NAK limit set in USB\_NAKLIMIT0 is reached. If the target function responds with a STALL, the USB controller does not retry the transaction, but does interrupt the processor core with the RXSTALL\_TH bit in the USB\_TXCSR register set. If the target function does not respond to the OUT token within the required time (or there was a CRC or bit-stuff error in the packet), the USB controller retries the transaction. If after three attempts the target function still has not responded, the USB controller flushes the FIFO and interrupts the processor core with the ERROR\_TH bit in USB\_TXCSR set.

## Multi-point Support

The following sections describe the controller's multi-point support

### Allocating Devices to Endpoints

The separate functions of the connected devices are allocated to the endpoints within the USB controller through a group of three registers, which are associated with each Rx or Tx endpoint implemented (including Endpoint 0).

The registers concerned are Tx/RxFuncAddr, Tx/RxHubAddr and Tx/RxHubPort. (The location of these registers depends on which of the endpoints is being addressed.)

The information that needs to be recorded in the Tx/RxFuncAddr register is the address of the target function that is to be accessed through the selected endpoint. This information needs to be recorded separately for each Tx and Rx endpoint that is used. In particular, both TxFuncAddr and RxFuncAddr need to be set for Endpoint 0.

The Tx/RxHubAddr and Tx/RxHubPort registers are provided for the case where a full- or low-speed device is connected to the USB controller via a high-speed USB 2.0 hub, which carries out the required transaction translation between high-speed transmission and low-/full-speed transmission. Where this is the case, the Tx/RxHubAddr and Tx/RxHubPort registers need to record the address of the hub that carries out the transaction translation and the port of that hub through which the associated Tx/Rx endpoint needs to access the device. Note that if Endpoint 0 is connected to a hub, then both the Tx and the Rx versions of these registers need to be set for this endpoint. The Tx/RxHubAddr register is further

used to record whether the hub offers multiple transaction translators or just a single transaction translator. This has a significant effect on the overall bandwidth that can be achieved.

In addition to recording the address of the target function through these three registers, the endpoint number and operating speed of the target device and the type of transaction that will be executed need to be recorded. For a Tx endpoint, this information needs to be set in the TxType register when the Index register is set to select the required endpoint. For an Rx endpoint, this information needs to be set in the RxType register when the Index register is set to select the required endpoint. In both cases, the endpoint number is recorded in bits D3 – D0, the transaction type is selected through bits D5 – D4, and the operating speed is selected through bits D7 – D6.

In the case of Endpoint 0, just the speed needs to be set (this endpoint only having the facilities to handle Control transactions and therefore always being associated with a device Endpoint 0). This speed setting is made through bits D7 – D6 of the Type 0 register, which is located at 1Ah when the Index register is set to 0.

## Multi-point Operation

Once the allocation of functions to endpoints has been made and the operating speed of the target device recorded, as described above, most operations in a Multi-point set-up are no different from those for the equivalent actions where the core is attached to a single other device.

However, additional steps are required:

- Where the option of dynamically switching the allocation of functions to endpoints is taken (for example to allow a wider range of devices to be supported)
- Where the control packets normally associated with Endpoint 0 are handled through a different endpoint.

If dynamic allocation is used, it becomes essential for the user to keep track of the current data toggle state associated with the endpoint and with each of the devices that are allocated to that endpoint. Knowledge of this state is necessary to allow the user to select the correct data toggle state when the switch is made between one device and other. (This action is the programs responsibility because the core cannot determine what data toggle state is expected when a function is being switched in and out of use.)

The data toggle state can be switched from its current state by writing to the appropriate TxCSRL/RxCSRL register to set the Data Toggle Write Enable and Data Toggle bits that are included in these registers when the core is in Host mode.

Data Toggle Write Enable and Data Toggle bits are also included in CSR0 when the core is operating in Host mode. However, Control operations carried out through the core's Endpoint 0 should normally always leave the data toggle in the expected state.)

Where control packets are handled through an endpoint other than Endpoint 0, the user has additionally to prompt for each Setup token to be sent. This involves setting the SetupPkt bit that is included in the TxCSR register when the core is operating in Host mode, alongside the TkPktRdy bit. If the SetupPkt bit is not set, an OUT token will be sent.

Overall, the recommendation is to use the controller's Endpoint 0 to handle Control packets for all of the devices attached to the controller, and to switch the allocation of this endpoint as appropriate. The issue of sending the correct token is then taken care of, as is the issue of ensuring that the data toggle is correctly set for this endpoint.

Using a different endpoint for this function is possible, as described above, but there are further points to note:

- The control function must be allocated to an Rx/Tx endpoint pair (i.e. with the same endpoint number).
- The chosen endpoints must each be associated with FIFOs that can accommodate the packet size associated with EP0 transactions at the chosen operating speed which can be a minimum of 8 bytes for low- or full-speed transactions but 64 bytes for high-speed transactions.

## Multi-point Bandwidth Considerations

The ability of a multi-point system to cope with isochronous transactions, in particular, is determined by the available bandwidth.

Once an endpoint has been set up, all scheduling is handled in hardware. However, as with PC-based EHCI/OHCI/UHCI hosts, before opening a periodic pipe (for use by isochronous or interrupt traffic), software must determine that there is sufficient bandwidth available. Further, if the periodic pipe is opened to a full-speed device through a high-speed hub, software must confirm that sufficient bandwidth is available both on the local high-speed bus and the full-speed bus generated by the transaction translator in the hub. The bandwidth required for different transactions can be determined using similar algorithms to those used in connection with PC-based hosts (detailed in Section 5.11.3 of the USB 2.0 Specification).

As would be expected, the bandwidth available will be greater where the hub used supports multiple transaction translators.

## Babble Interrupt

If the bus is still active at the end of a frame, the USB controller assumes that the function it is connected to has malfunctioned, suspends all transactions, and generates a babble interrupt (`RESET_OR_BABLE_B`). The USB controller does not start a transaction until the bus is inactive for at least the minimum inter-packet delay. The controller also does not start a transaction unless it can be finished before the end of the frame.

To recover from a babble error condition, the processor must take the following actions inside the interrupt service routine.

1. Turn off VBUS. Wait until the VBUS level indicator reads b#01.
2. Turn on VBUS. Wait until the VBUS level indicator reads b#11.
3. Set the SESSION bit



The VBUS level indicator is a field of the USB\_OTG\_DEV\_CTL register.

**NOTE:** Because VBUS is sourced external to the processor, make sure that in your hardware design you connect a GPIO or the dedicated `DrvVBUS` signal to the external source so that you can use software to turn VBUS on and off.

## VBUS Events

The USB On-The-Go specification defines a series of thresholds to which the devices involved in point-to-point communications are required to respond.

- VBUS Valid (required to be between 4.4V and 4.75V)
- Session Valid for A device (required to be between 0.8V and 2.1V)
- Session End (required to be between 0.2V and 0.8V)

Which thresholds are critical and the processor response depends upon whether the device is an A device or a B device and the circumstances of the event. These actions are described below.

### Actions as an “A” Device

VBUS >VBUS Valid with session initiated by USB controller (that is, VBUS level indicator = b#11 and session bit is set). When VBUS is greater than VBUS Valid, the USB controller selects Host Mode and waits for a device to be connected. It then generates a connect interrupt. The processor resets and enumerates the connected B device.

VBUS> Session Valid with session initiated by B device (that is, VBUS level indicator = b#10 and session bit is clear). When VBUS is greater than Session Valid, the USB controller generates a session request interrupt. The processor sets the session bit and the USB controller either stays in Host mode or changes to Peripheral mode, depending upon the state of the pull-up resistor on the B device. For more information, refer to the Host Negotiation Protocol of the OTG specification. The selected mode is indicated by the state of the Host Mode bit.

VBUS below VBUS Valid while the Session bit remains set (that is, VBUS level indicator b#11 and session bit is set). This indicates a problem with the VBUS power level. For example, the battery power may have dropped too low to sustain VBUS Valid. Or, the B device may be drawing more current than the A device can provide. In either case, the USB controller will automatically terminate the session and generate a VBUS error interrupt.

To recover from this VBUS error condition, the processor must take the following actions inside the VBUS error interrupt handler.

- Turn off VBUS Wait until the VBUS level indicator reads b#01.
- Turn on VBUS Wait until the VBUS level indicator reads b#11.
- Set the SESSION bit

The VBUS level indicator is a field of the USB\_OTG\_DEV\_CTL register.

**NOTE:** Because VBUS is sourced external to the processor, make sure that in your hardware design you connect a GPIO or the dedicated `DrvVBUS` signal to the external source so that you can use software to turn VBUS on and off.

## Actions as a “B” Device

`VBUS > Session Valid` (that is, VBUS level indicator = b#10 and session bit is clear). This indicates activity from the A device. The USB controller sets the session bit and disconnects the pull down resistor on the D+ line.

`VBUS < Session Valid` while the session bit remains set (that is, VBUS level indicator = b#01 and session bit is set). This indicates that the A device has lost power (or become disconnected). The USB controller clears the session bit and generates a disconnect interrupt. The processor ends the session.

`VBUS < Session End` (that is, VBUS level indicator = b#00). This is the condition under which a B device can initiate a session request. If the session bit is set, then after 2ms of SE0 on the bus, the USB controller starts SRP by first pulsing the data line, then pulsing VBUS.

## Host Mode Reset

If the `RESET` bit in the `USB_POWER` register is set while the USB controller is in host mode, the USB controller generates reset signaling on the bus. The processor core should keep this bit set for 20 ms to ensure correct resetting of the target device. After the processor core has cleared the bit, the USB controller starts its frame counter and transaction scheduler.

## Host Mode Suspend

**Entry into Suspend mode.** When operating as a host, the USB controller can be prompted to go into Suspend mode by setting the Suspend Mode bit in the Power register. When this bit is set, the USB controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated. If the `Enable SuspendM` bit is set, the UTMI+ PHY will go into low-power mode when the USB controller goes into Suspend mode and stop XCLK.

**Sending Resume Signaling.** When the application requires the USB controller to leave Suspend mode, it needs to clear the Suspend bit in the Power register, set the Resume bit and leave it set for 20ms. While the Resume bit is high, the USB controller will generate Resume signaling on the bus. After 20 ms, the processor core should clear the Resume bit, at which point the frame counter and transaction scheduler will be started.

**Responding to Remote Wake-up.** If Resume signaling is detected from the target while the USB controller is in Suspend mode, the UTMI+ PHY will be brought out of low-power mode and restart XCLK. The USB controller will then exit Suspend mode and automatically set the Resume bit in the Power register to ‘1’ to take over generating the Resume signaling from the target. If the Resume interrupt is enabled, an interrupt will be generated.

## Suspending and Resuming the Controller

With the introduction of Link Power Management, there are two basic methods for the USB controller to be suspended and resumed. These two methods are demonstrated in the basic LPM transaction diagram shown below.

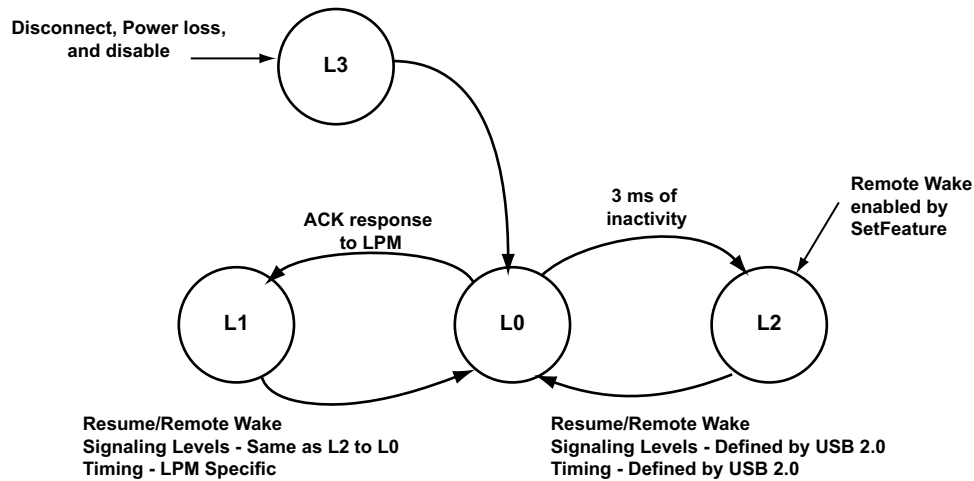


**Figure 22-9:** Basic LPM Transaction

The procedure in which the USB controller is suspended and resumed depends on whether the core is operating as a device or a host and the method of suspend desired. These options are described in the following sections.

### Suspend/Resume By Inactivity On The USB Bus (L0 To L2 State) In Peripheral Mode

The following steps occur in this mode.

1. Entry into Suspend mode. When operating as a peripheral, the USB controller monitors activity on the USB and when no activity has occurred for 3 ms, it goes into Suspend mode. If the Suspend interrupt has been enabled, an interrupt will be generated at this time. The SUSPENDM output will also go low (if enabled).

At this point, the POWERDWN signal is also asserted to indicate that the application may save power by stopping CLK. POWERDWN then remains asserted until either power is removed from the bus (indicating that the device has been disconnected) or Resume signaling or Reset signaling is detected on the bus.

2. When Resume signaling occurs on the bus, first CLK must be restarted if necessary. The USB controller will then automatically exit Suspend mode. If the Resume interrupt is enabled, an interrupt will be generated.
3. Initiating a Remote Wakeup. If the software wants to initiate a remote wakeup while the controller is in Suspend mode, it should write to the USB\_Power register to set the Resume bit 2 to 1. (Note: If CLK has been stopped, it will need to be restarted before this write can occur.) The software should leave

then this bit set for approximately 10 ms (minimum of 2 ms, a maximum of 15 ms) before resetting it to 0. By this time the hub should have taken over driving Resume signaling on the USB.

**NOTE:** No Resume interrupt will be generated when the software initiates a remote wakeup.

### **Suspend/Resume By Inactivity On The USB Bus (L0 To L2 State) In Host Mode**

The following steps occur in this mode.

1. Entry into Suspend mode. When operating as a host, the USB controller can be prompted to go into Suspend mode by setting the Suspend Mode bit in the Power register. When this bit is set, the USB controller will complete the current transaction then stop the transaction scheduler and frame counter. No further transactions will be started and no SOF packets will be generated. If the Enable SuspendM bit (Power.D0) is set, the UTMI+ PHY will go into low-power mode when the controller goes into Suspend mode and stop XCLK.
2. Sending Resume Signaling. When the application requires the controller to leave Suspend mode, it needs to clear the Suspend bit in the Power register, set the Resume bit and leave it set for 20 ms. While the Resume bit is high, the controller will generate Resume signaling on the bus. After 20 ms, the processor core should clear the Resume bit, at which point the frame counter and transaction scheduler will be started.
3. Responding to Remote Wake-up. If Resume signaling is detected from the target while the USB controller is in Suspend mode, the UTMI+ PHY will be brought out of low-power mode and restart XCLK. The controller will then exit Suspend mode and automatically set the Resume bit in the USB\_Power register to 1 to take over generating the Resume signaling from the target. If the Resume interrupt is enabled, an interrupt will be generated.

### **Suspend/Resume By an LPM Transaction (L0 To L1 State) In Peripheral Mode**

The following steps occur in this mode.

1. Entry into Suspend mode. When operating as a peripheral, the controller will never initiate an LPM Suspend (transition from the L0 state to the L1 state). Rather, the controller will only Suspend at the request of the host. However, for this to occur, the LPM feature must be enabled by setting up the LPM\_CNTRL register (address 0x362) appropriately. The register field LPMEN (in the LPM\_CNTRL register) is used to enable and support extended and LPM transactions. The LPMXMT field is used instruct the hardware that it is ready to suspend and to respond to the next LPM transaction with an

ACK. In this case, the controller will respond to the next LPM transaction with an ACK if all other conditions are met. The response to an LPM transaction by the controller is summarized below:

**Table 22-6:** Response to LPM Transaction

| LPMXMT                        | LPMCNTL                           | Data Pending (Resides in Tx FIFOs) | Response to Next LPM Transaction |
|-------------------------------|-----------------------------------|------------------------------------|----------------------------------|
| 1'b0,<br>1'b0<br>1'b1<br>1'b1 | 2'b00,<br>2'b10<br>2'b00<br>2'b10 | Don't Care                         | Timeout                          |
| 1'b0,<br>1'b1                 | 2'b01                             | Don't Care                         | STALL                            |
| 1'b0                          | 2'b11                             | Don't Care                         | NYET                             |
| 1'b1                          | 2'b11                             | Yes                                | NYET                             |
| 1'b1                          | 2'b11                             | No                                 | ACK                              |

For all cases shown above in which the controller responds (No timeout occurs), an LPM interrupt will be generated in the LPM\_INTR register. Note that the controller will respond with an ACK only if there is no data pending in any of the TX Endpoint fifos. If there is data pending, the USB controller will respond with a NYET.

Once an LPM transaction is successfully received three events will occur:

- The LPM\_ATTR register (address 0x360) will be updated with values received in the LPM transaction just received. The LPM\_ATTR register contains the following fields:

**LinkState** - This field tells the controller what state to transition to. The only valid value for this field is 4'h1 indicating that the core should suspend. For any other value, the MUSBMHDCR will respond with a STALL and the appropriate interrupt will be generated. However, in this case the LPM\_ATTR register will be updated so that software can observe the non compliant LPM packet payload. In addition, the LPMERR interrupt will also be generated informing software of the non-compliant LPM transaction.

**HIRD** - This field tells the controller the minimum duration that the host will drive resume signaling on the bus. This field represents a resume duration range from 50us to 1200us. This value may be different for subsequent LPM transactions.

**RmtWak** - This is a 1 bit field indicating if the remote wakeup by the controller is allowed. This bit only applies to the current suspend/resume cycle only. This bit may be different for subsequent LPM transactions. This bit should not supersede the wakeup capability that was previously negotiated on enumeration of the controller.

- b. The controller will suspend 9us after transmitting the ACK. Resume signaling can be driven by the Host or the controller 50us after this event. During this 9us interval, the host may continue to transmit the LPM transaction. The controller will respond with an ACK in this case regardless of the LPMXMT value in the LPM\_CNTRL register.
- c. An interrupt will be generated informing software of the response (An ACK in this case). An ACK response is the indication to software that the controller has suspended.

Since the primary purpose of LPM is to save power, the software will read the LPM\_ATTR register to determine the attributes of the Suspend. Software must make a determination based on these attributes whether additional power savings in the system can be exploited. In making this determination it is noted that if the host initiates the resume signaling, the controller will be required to respond to packet transmissions within the time specified by HIRD + 10 us.

2. When resume signaling occurs on the Bus When the host resumes the bus, it will drive resume signaling for a minimum time specified by the HIRD field in the LPM\_ATTR register. The controller must be able to respond to traffic within the time HIRD + 10us. The controller will transition to a normal operating state automatically and a resume interrupt will be generated in the LPM\_INTR register. However for this to occur, the inputs CLK and XCLK must be available. To facilitate the resume timing requirement, an additional feature, LPMNAK, is provided in the LPM\_CNTRL register. If LPMNAK is set to 1'b1, all endpoints will respond to any transaction (other than an LPM) with a NAK. This bit will only take effect after the controller has LPM suspended. Typically, this bit would be asserted when the LPMXMT field is also asserted. Using this feature may simplify the resume timing requirement because only XCLK is needed for the controller to respond (with a NAK) to traffic. Software can continue to restore the system to normal operation while the controller responds to all transactions with a NAK. After the system has been completely restored, software can then clear the LPMNAK field in the LPM\_CNTRL register.
3. Initiating remote wakeup. If the software wants to initiate a remote wakeup while the controller is in Suspend mode, it should write a 1'b1 to the LPMRES bit in the LPM\_CNTRL register. This bit is self clearing. Writing a 1'b1 will cause resume signaling to be driven on the bus for 50us. The host will respond by driving resume for 60us to 990us. 10 us after the host stops driving resume, the controller will transition to its normal operational state and will be ready for packet transmission. A Resume interrupt will be generated in the LPM\_INTR register.

### **Suspend/Resume by an LPM Transaction (L0 to L1 State) in Host Mode**

The following steps occur in this mode.

1. Entry into Suspend mode. When operating as a host, the controller will initiate an LPM Suspend (transition from the L0 state to the L1 state) by initiating an LPM transaction as follows:
  - a. Software will set-up the desired attributes of the Suspend in the LPM\_ATTR register (address 0x360). Enabling remote wakeup and a large HIRD will give the peripheral more opportunity to conserve power.
  - b. All LPM interrupts should be enabled in the LPM\_INTEN register.

- c. Software should initiate the transaction by writing a 0x01 to the LPM\_CNTRL register.
- d. An interrupt is generated to inform software of the response to the LPM transaction. If an ACK was received, then the controller will suspend automatically within 8us. This is the indication that the controller has suspended.

If the response from the device has a bit stuff error or a PID error, then an LPMERR interrupt is generated. The hardware will immediately attempt the LPM transaction two more times. The device will not suspend for 8us after the initial LPM so it will be able to respond to either of these subsequent LPM transactions. If an LPM timeout has occurred three times, the LPMNC and the LPMERR interrupts will be set. At this time, software is unaware of the device state and must deduce it by other means.

2. Sending Resume Signaling. Resume signaling should be generated by software as follows:
  - a. All LPM interrupts should be enabled in the LPM\_INTEN register.
  - b. Software should write the LPMRES bit in the LPM\_CNTRL register. This bit is self-clearing. This will cause resume signaling to be generated on the Bus for the time that is currently specified in the HIRD field in the LPM\_ATTR register. It is assumed by hardware that this value was used in the last LPM transaction that caused the Suspend.
  - c. After HIRD + 10us, the controller will transition to its normal operational state and be ready for packet transmission. A resume interrupt will be generated in the LPM\_INTR register.

**NOTE:** Prior to resuming, software must ensure that the system is completely restored from a low power state and that the inputs CLK and XCLK are available.

3. Responding to Remote Wake-up. If the remote wakeup feature was enabled in the LPM transaction that caused the Suspend, then the device may drive resume signaling on the bus. When this occurs, the device will drive resume signaling BUS for 50us. The controller will immediately begin driving resume signaling on the BUS and will do so for 60us. 10us after completion of the resume signaling, the controller will transition to its normal operating state and will be ready for packet transmission. At this time, the resume interrupt is generated in the LPM\_INTR register.

## USB Event Control

The following sections provide information on the use of interrupts, reset and the reporting of errors and interface status.

## USB Interrupt Signals

There are two interrupts generated from the USB Controller:

1.  $\text{INTR\_USB0\_STAT} = !(\text{USB0\_INTRTX} \mid \text{USB0\_INTRRX} \mid \text{USB0\_IRQ} \mid \text{DrvVbusInt}) \mid \text{USB0\_LPM\_IRQ}$
2.  $\text{INTR\_USB0\_DATA} = \text{USB0\_DMA\_IRQ}$

Interrupts can be generated from control endpoint zero under the following conditions

- When a control transaction ends before the end of the data is transferred.
- When a data packet is sent or received from the endpoint 0 FIFOs.

Interrupts can be generated from transmit endpoints (USB\_INTRTX) under the following conditions:

- packet sent from the TX FIFO (host and peripheral mode)
- after three attempts at transmitting a packet with no valid handshake packet received (host mode)

Interrupts can be generated from receive endpoints (USB\_INTRRX) under the following conditions:

- packet received into the RX FIFO (host and peripheral mode)
- when a STALL handshake is received (host mode)
- After three attempts at receiving a packet and no data packet is received (host mode).

Interrupts can be generated from the USB status (USB\_INTRUSB) under the following conditions:

- When VBUS drops below the VBUS valid threshold during a session (A device only).
- When SRP signaling is detected (A device only).
- When device disconnect is detected (host mode).
- When a session ends (peripheral mode).
- Device connection detected (host mode).
- Start-of-frame (SOF)
- Reset signaling detected on USB (peripheral mode).
- Babble detected (host mode).
- In suspend mode when resume signaling detected on USB.
- When suspend signaling is detected (peripheral mode).

Interrupts are generated for the following VBUS control requests by the USB controller:

- drive VBUS greater than 4.4V (Default A device)
- stop driving VBUS
- start charging VBUS (peripheral mode)
- stop charging VBUS
- start discharging VBUS (peripheral mode)
- stop discharging VBUS



## Interrupt Handling

When the processor core is interrupted with a USB interrupt, it needs to read the interrupt status register to determine which endpoint(s) have caused the interrupt and jump to the appropriate routine. If multiple endpoints have caused the interrupt, Endpoint 0 should be serviced first, followed by the other endpoints. A flowchart for the USB Interrupt Service Routine is shown below.

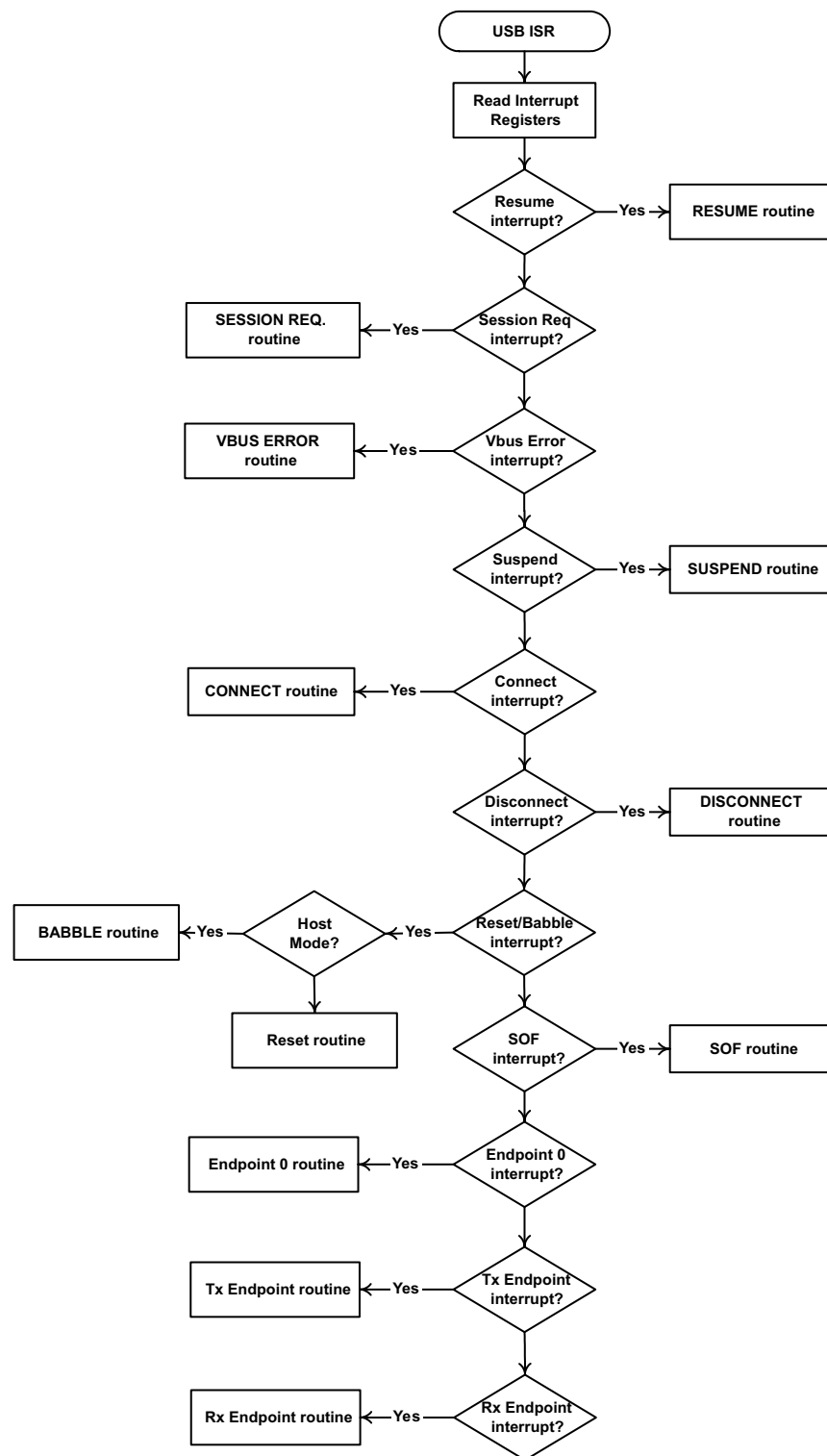


**Figure 22-10:** USB Interrupt Service Routine

## USB Reset Signals

The USB controller includes an active-high synchronous hardware reset sourced from the processor core. Another source of peripheral reset is through the USB, when USB reset signaling is detected on the I/O lines. As dictated by the USB 2.0 Specification, this state is entered when both the D+ and D- inputs are driven low for a period of 2.5s or more (though the reset itself is held for typically greater than 10ms by the USB host).

### USB Reset in Peripheral Mode

When a USB reset is detected, the USB controller performs the following actions:

- `USB_FADDR` register set to zero
- `USB_INDEX` register set to zero
- all endpoint FIFOs flushed
- all control and status registers cleared
- all interrupts enabled
- reset interrupt generated

The `USB_INTRUSB` and `USB_OTG_VBUS_IRQ` registers are not affected by the USB controller reset. These registers are only reset (along with those listed above) during a system reset.

If the `HS_ENABLE` bit in the power register was set, the USB controller also tries to negotiate for high-speed operation. Whether high-speed operation is selected is indicated by `HS_Mode` bit in the Power register.

When the application software receives a reset interrupt, it should close any open pipes and wait for bus enumeration to begin.

### USB Reset in Host Mode

If the Reset bit in the power register is set while the USB controller is in host mode, the controller will generate Reset signaling on the bus. If the `HS_ENABLE` bit in the power register was set, it will also try to negotiate for high-speed operation.

The processor core should keep the Reset bit set for at least 20 ms to ensure correct resetting of the target device. After the processor core has cleared the bit, the USB controller starts its frame counter and transaction scheduler.

High-speed operation is selected by the `HS_MODE` bit

## USB Programming Model

The following sections describe the USB OTG programming model.

### Peripheral Mode Flow Charts

Host actions are shown white. USB actions are shaded

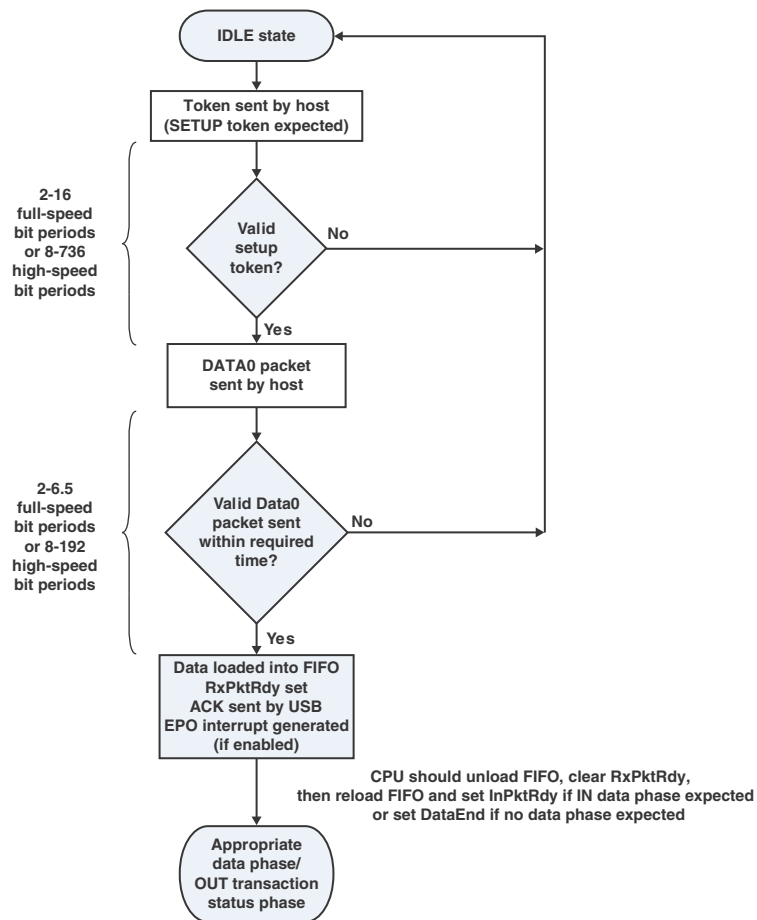


**Figure 22-11:** USB Control Setup Phase

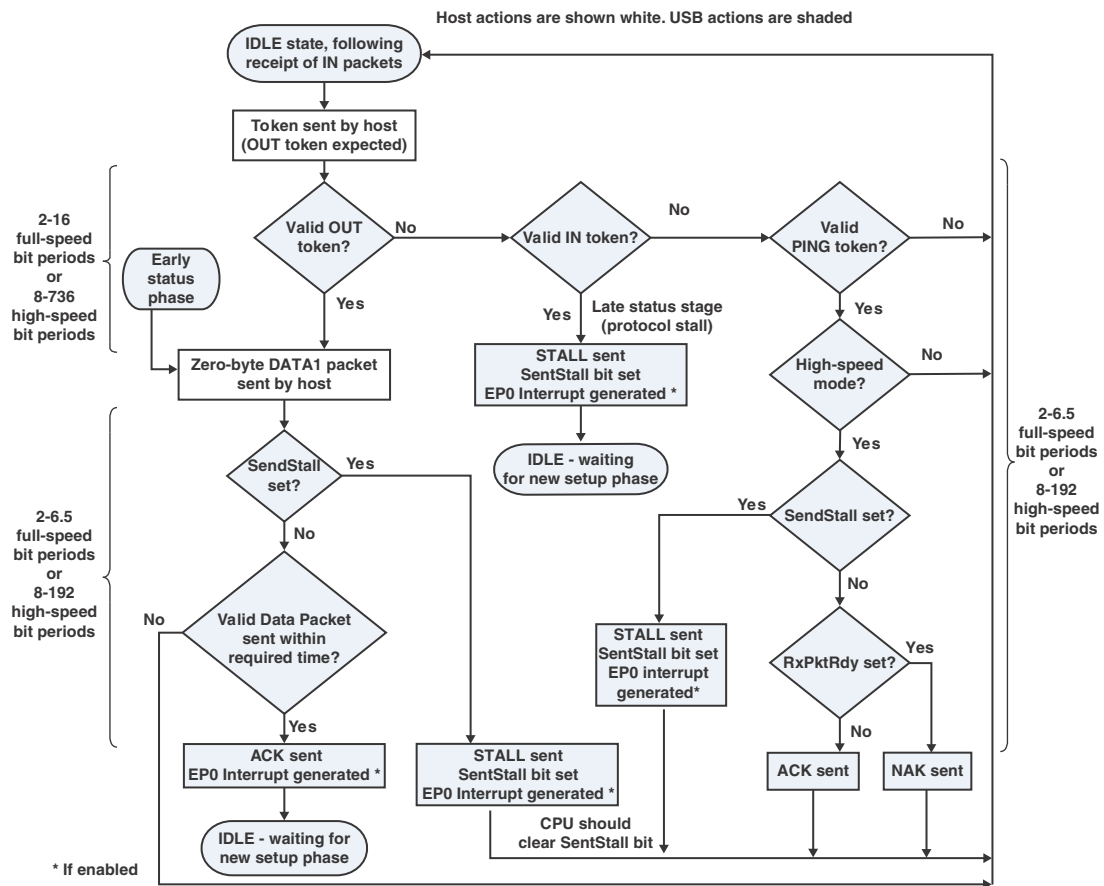


Figure 22-12: Control In Data Phase

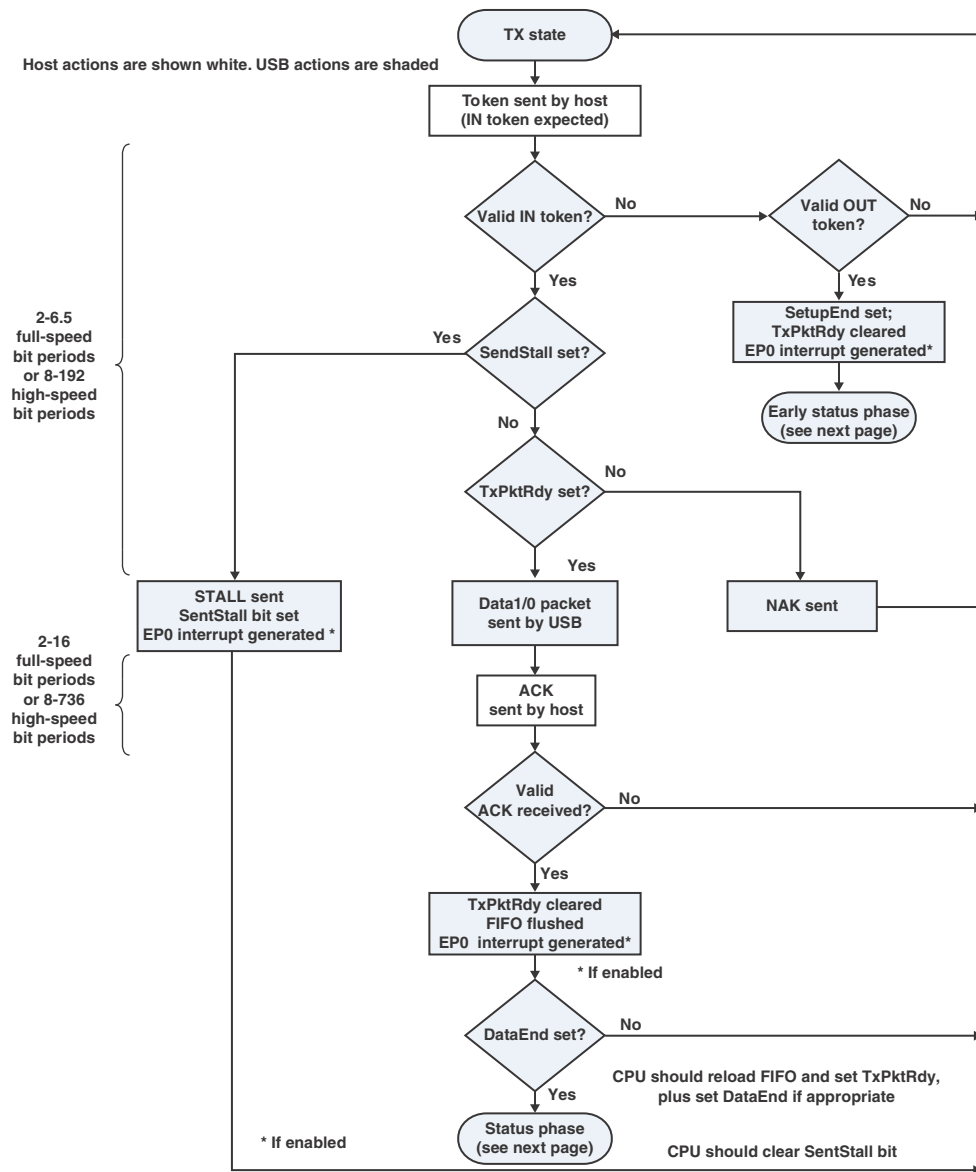


Figure 22-13: Control In Data Status Phase

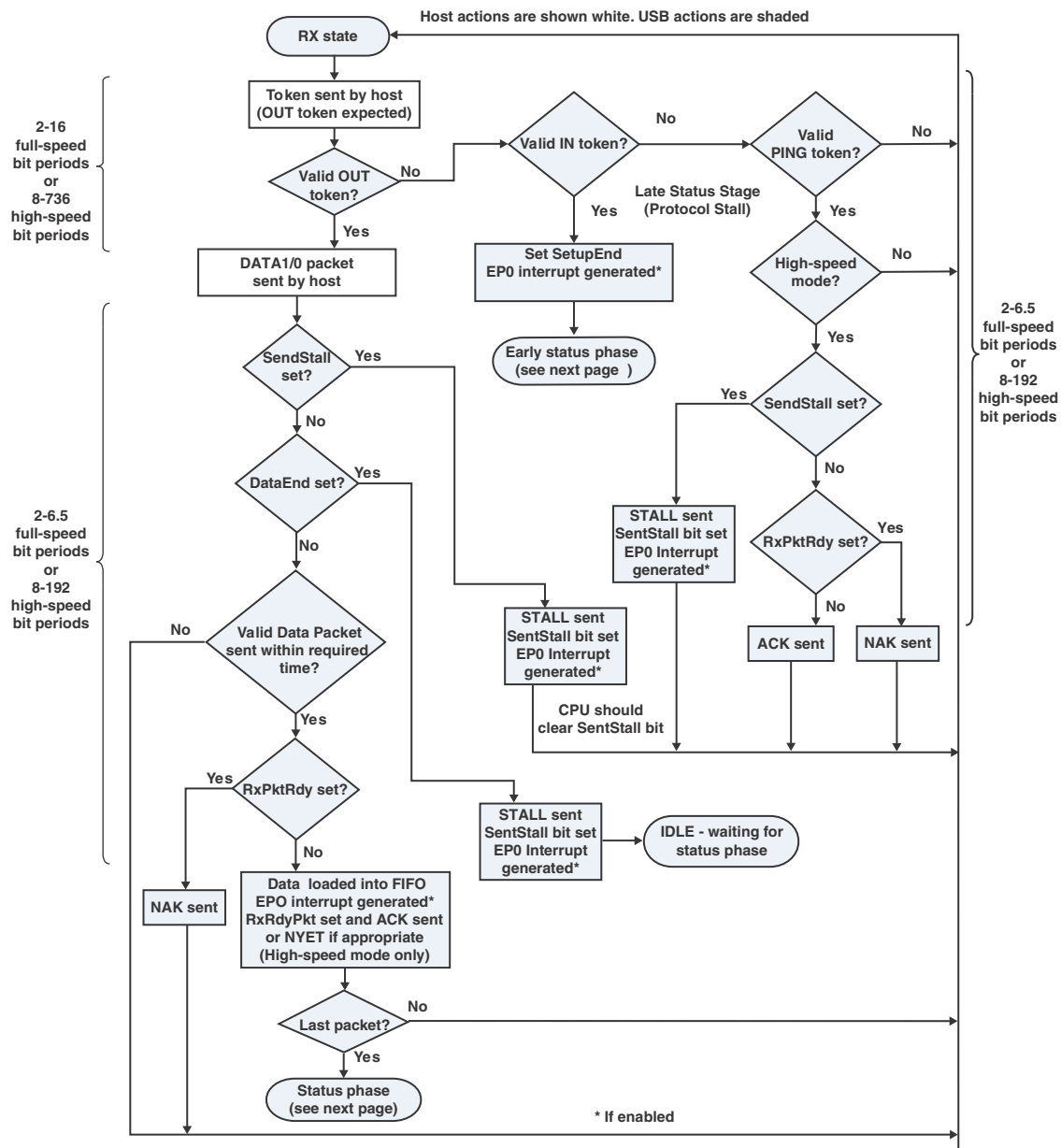


Figure 22-14: Control Out Data Phase

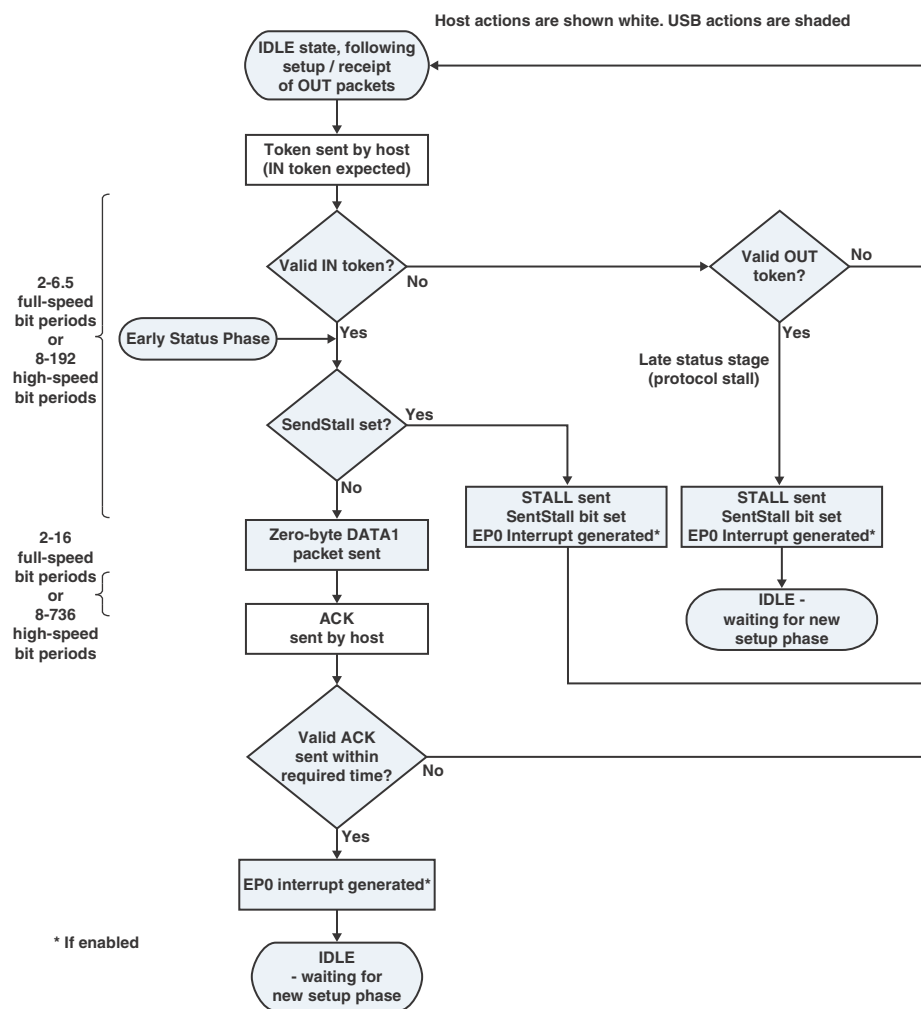


**Figure 22-15:** Control Out Data Status Phase



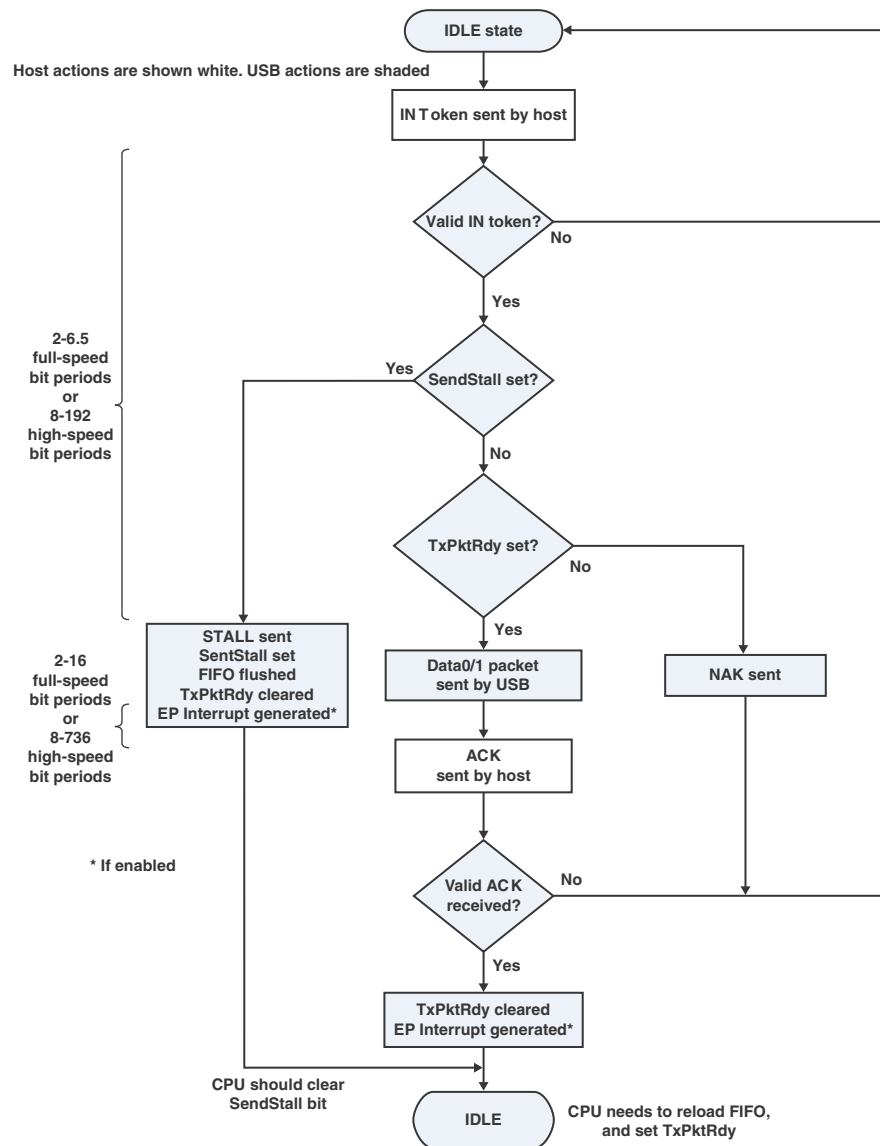


**Figure 22-16:** Bulk/Low Bandwidth Interrupt In Transaction

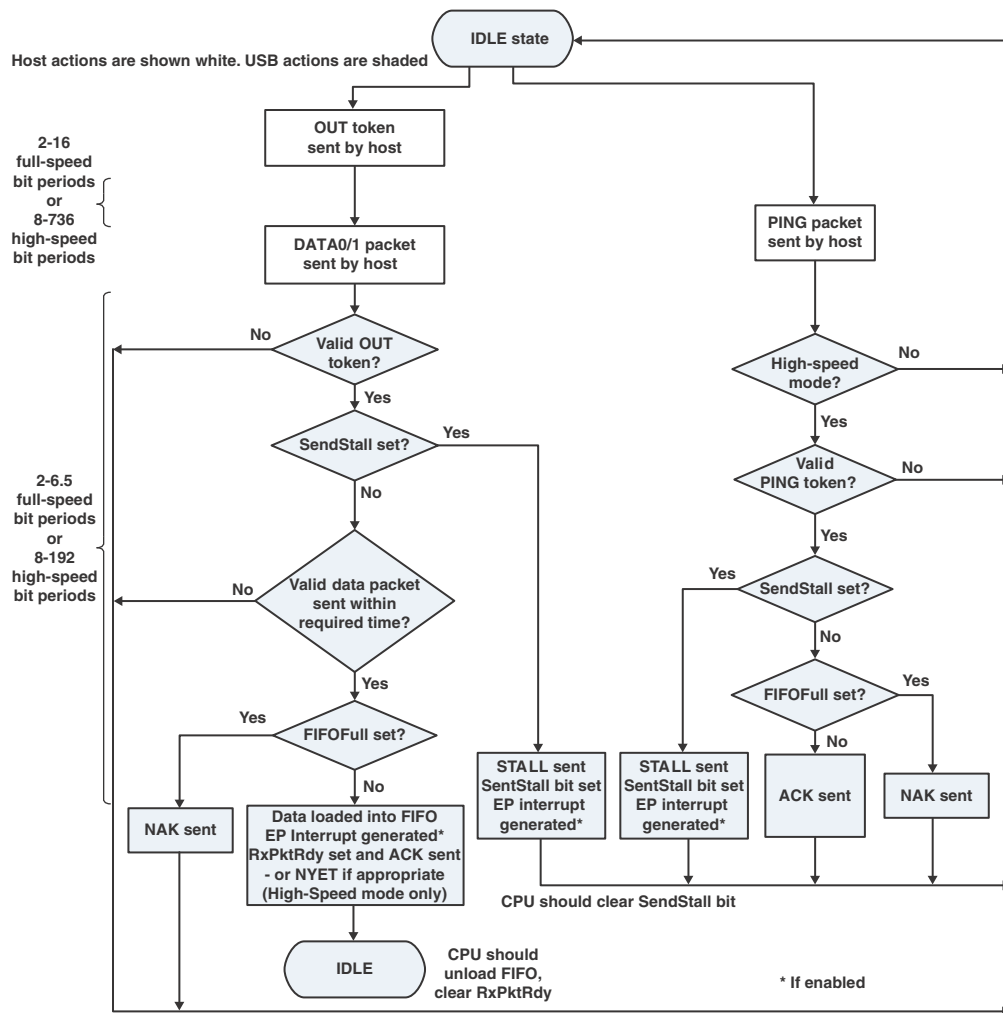


Figure 22-17: Bulk/Low Bandwidth Interrupt Out Transaction

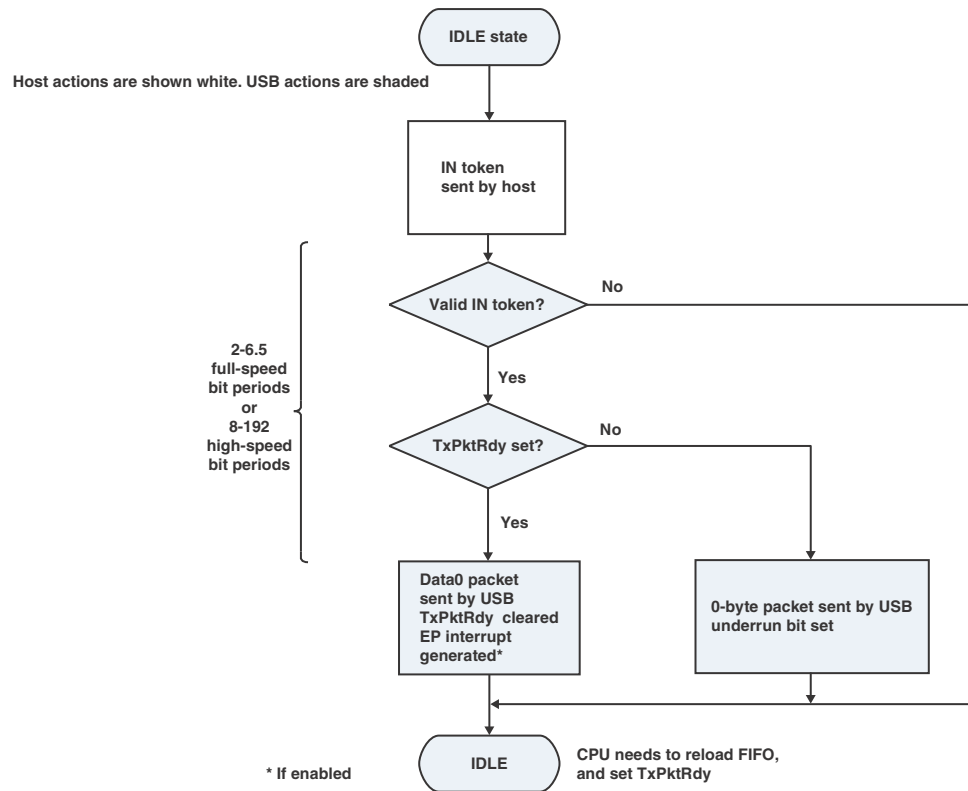


**Figure 22-18:** Full-speed/Low Bandwidth Isochronous In Transaction

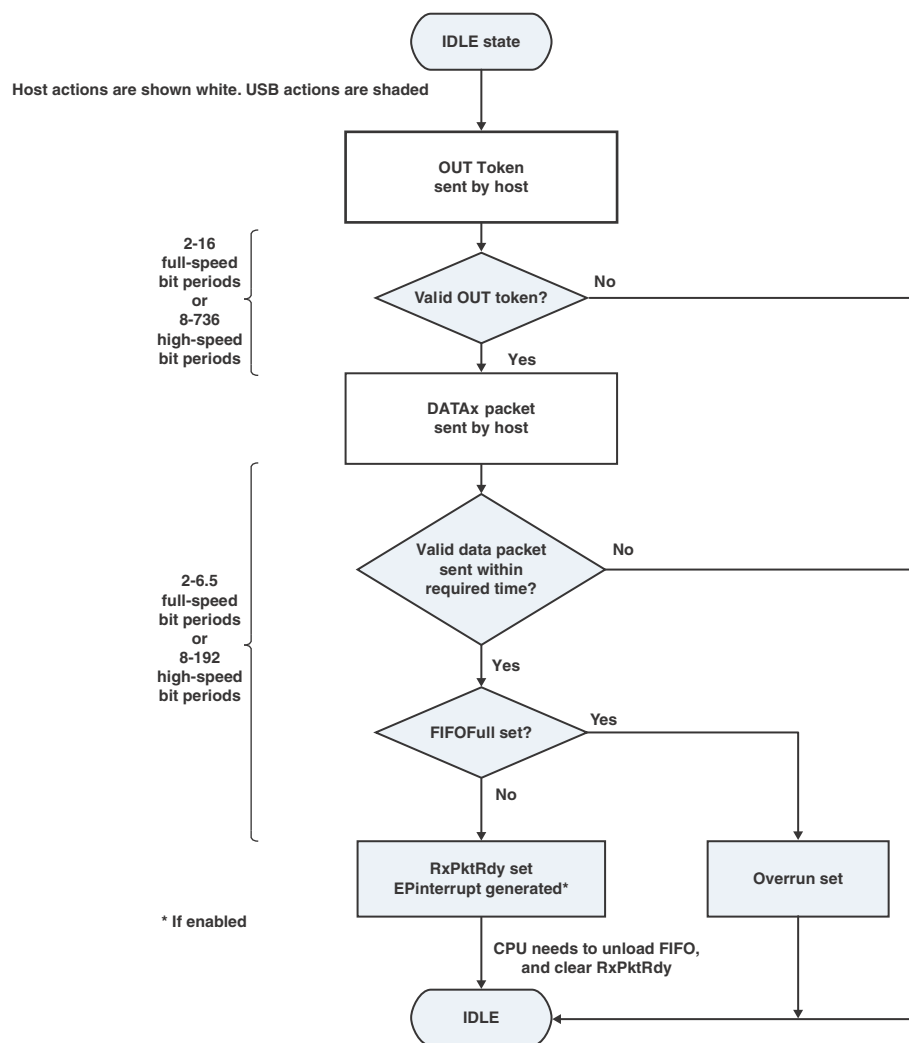


**Figure 22-19:** Full-speed/Low Bandwidth Isochronous Out Transaction

## Host Mode Flow Charts

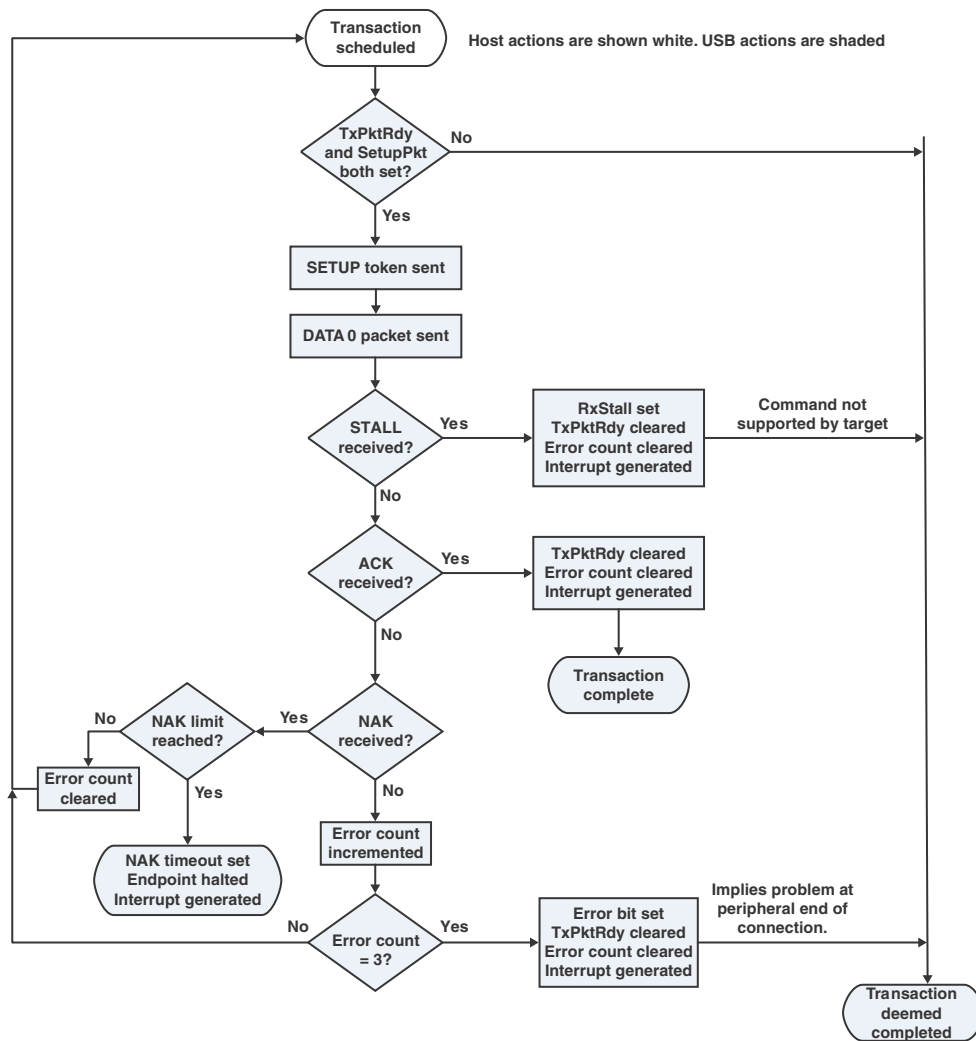


**Figure 22-20:** USB Control Setup Phase

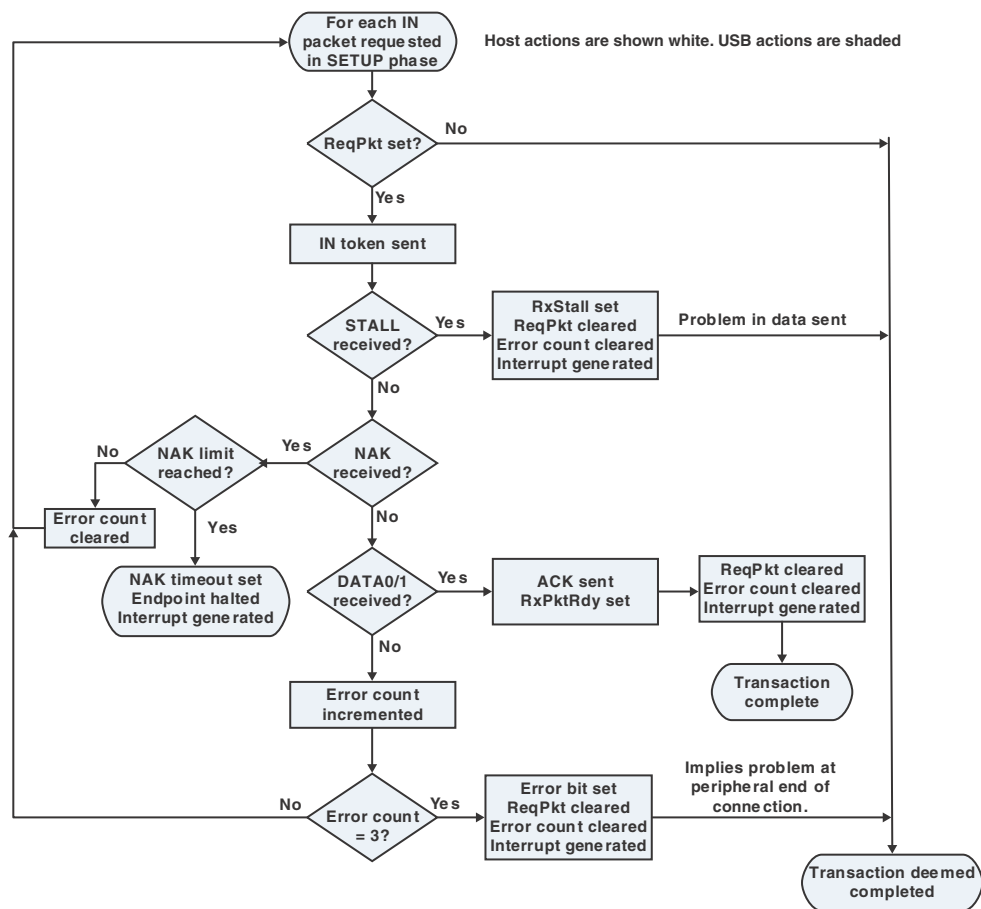


**Figure 22-21:** Control In Data Phase

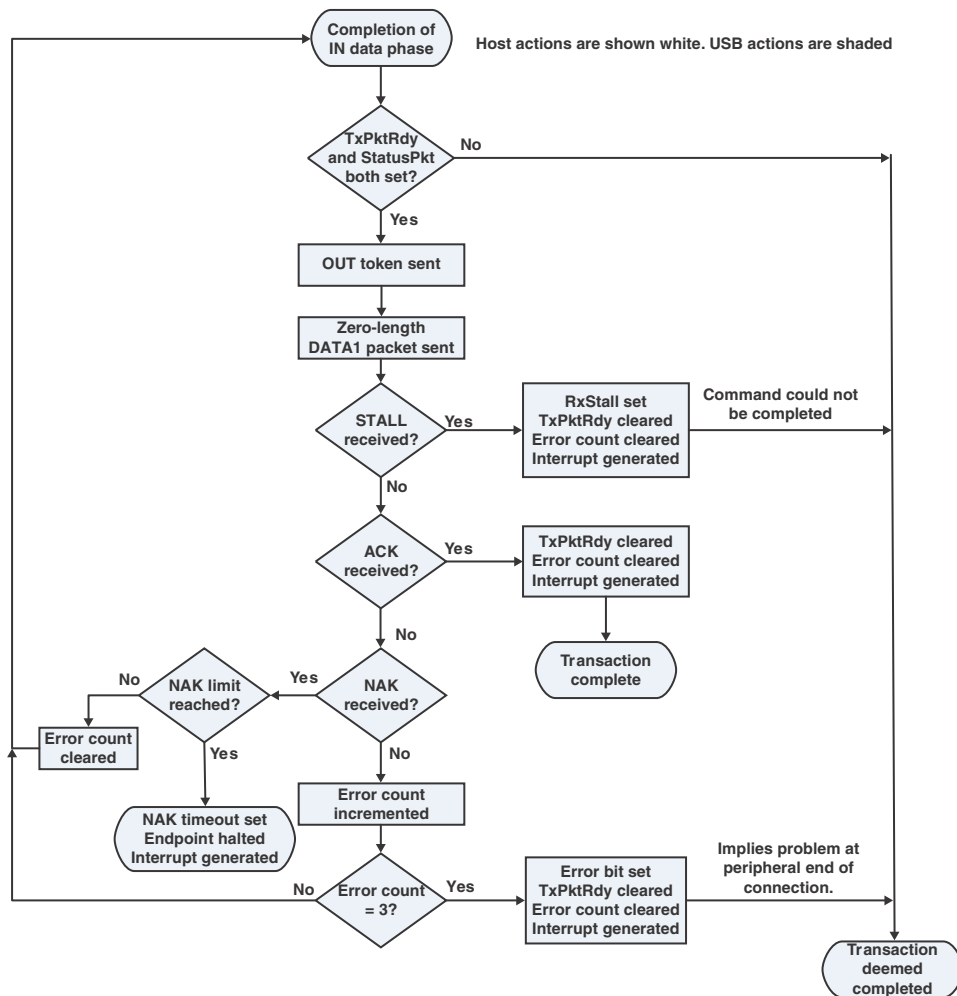


**Figure 22-22:** Control In Data Status Phase

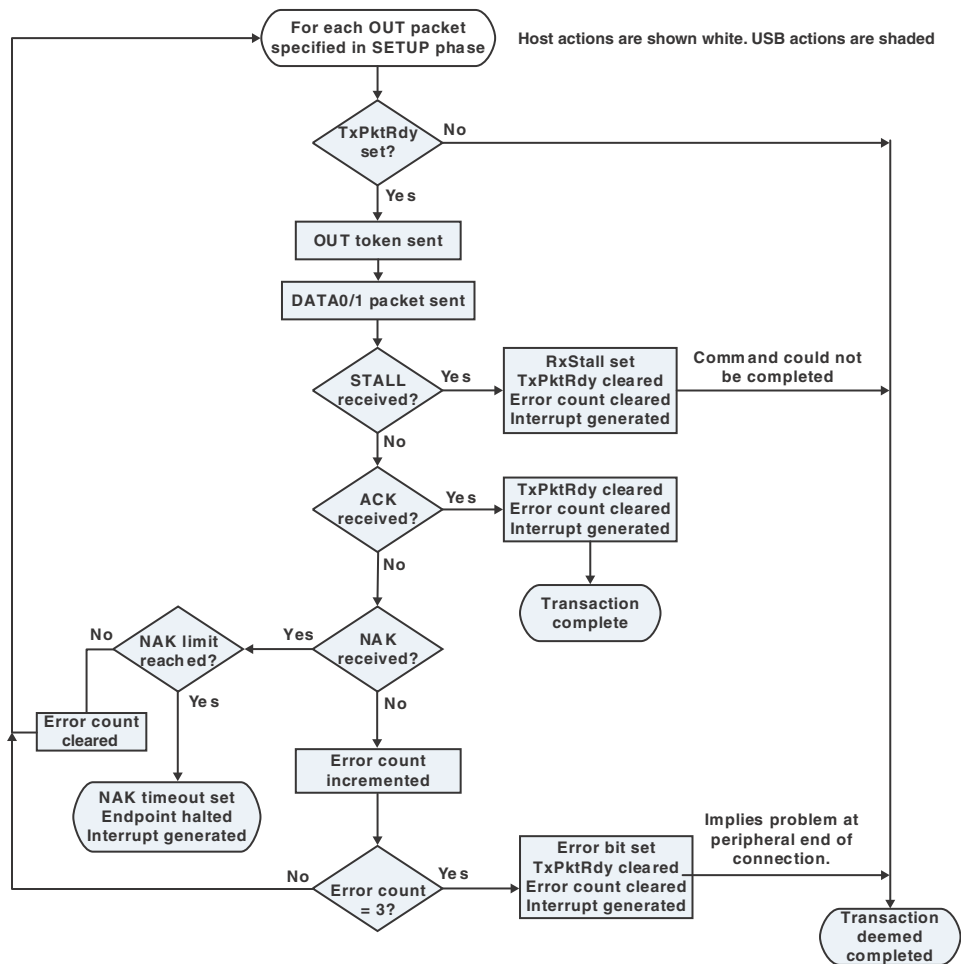


Figure 22-23: Control Out Data Phase



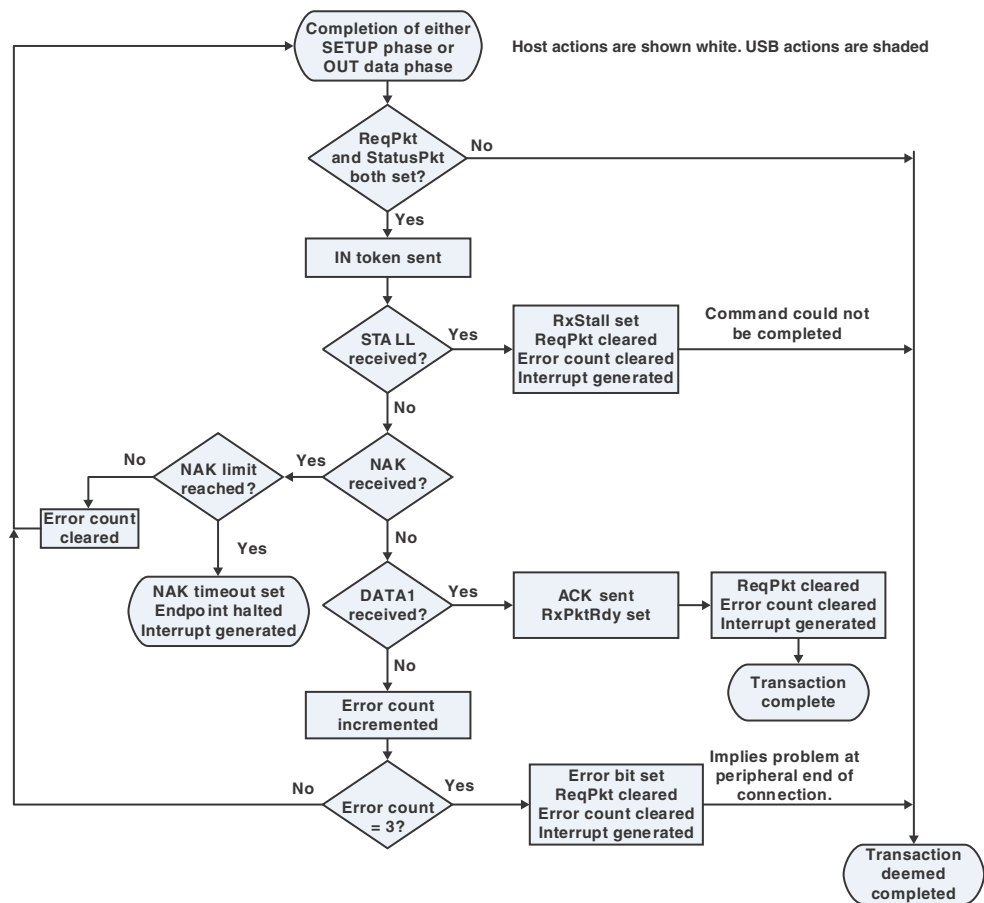


**Figure 22-24:** Control Out Data Status Phase

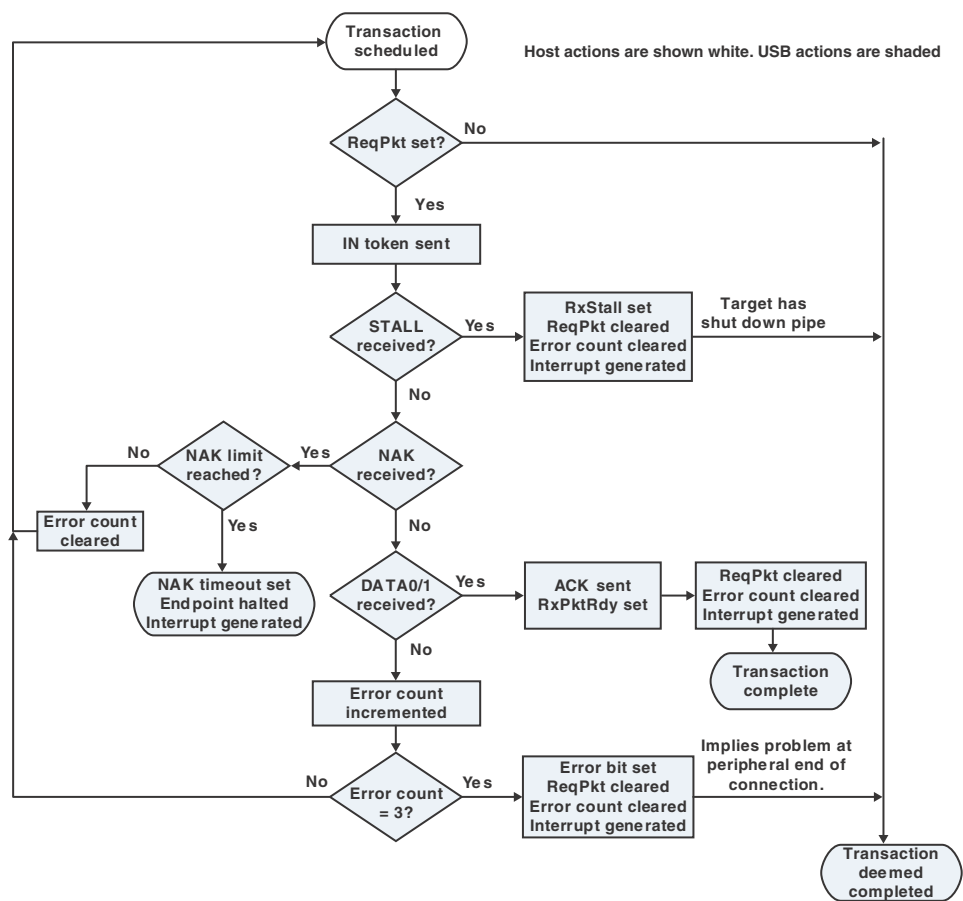


**Figure 22-25:** Bulk/Low Bandwidth Interrupt In Transaction

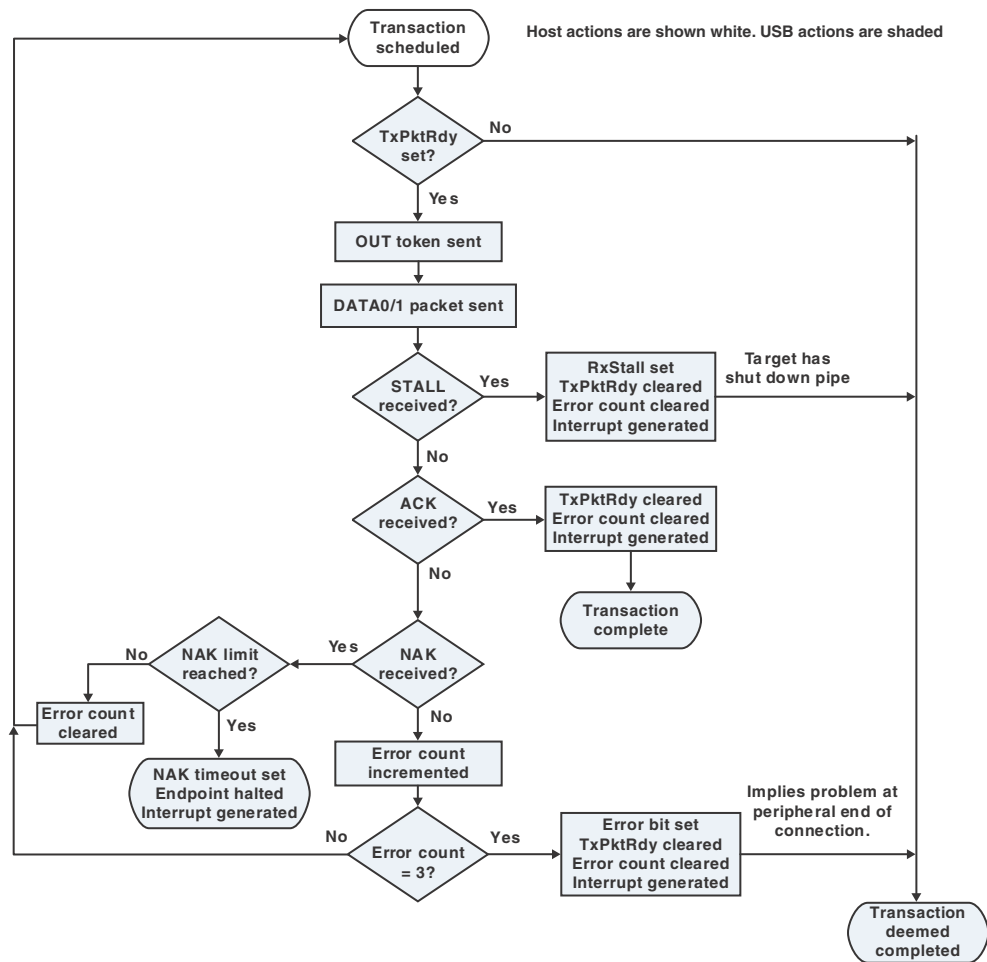


**Figure 22-26:** Bulk/Low Bandwidth Interrupt Out Transaction

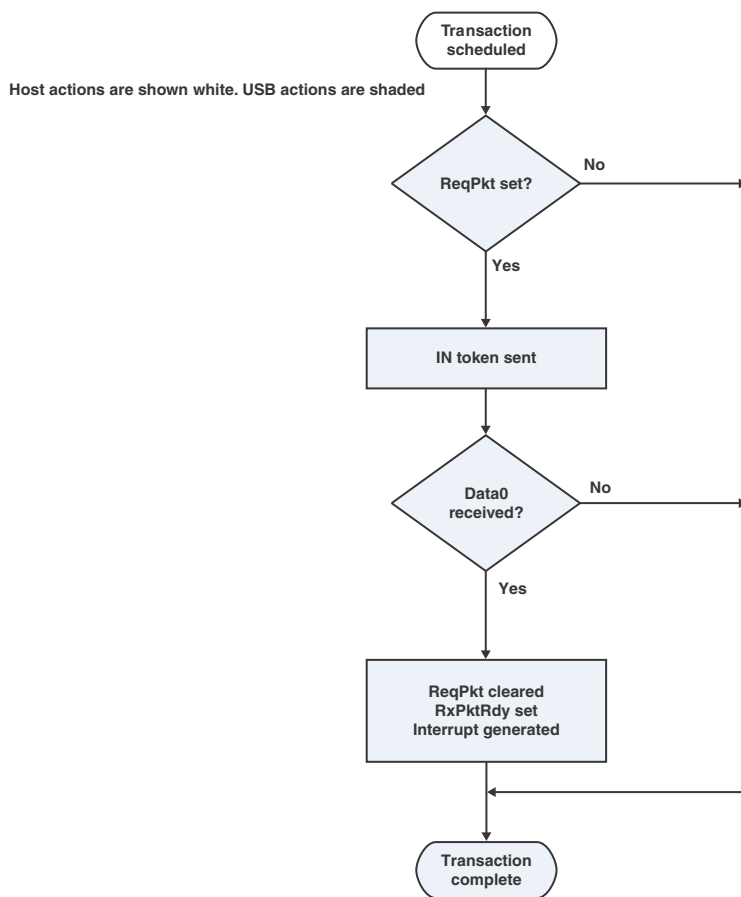


**Figure 22-27:** Full-speed/Low Bandwidth Isochronous In Transaction

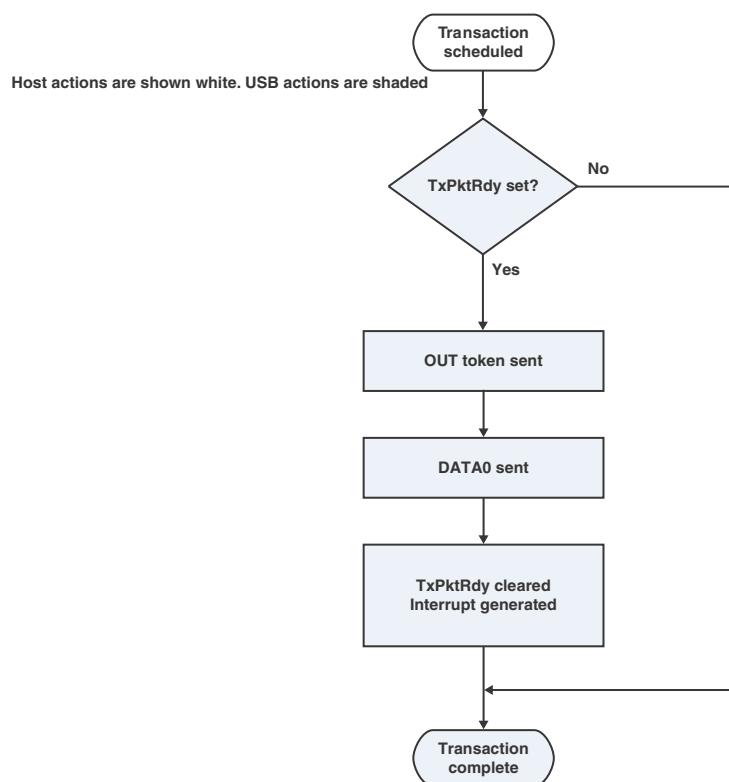


**Figure 22-28:** Full-speed/Low Bandwidth Isochronous Out Transaction

## DMA Mode Flow Charts

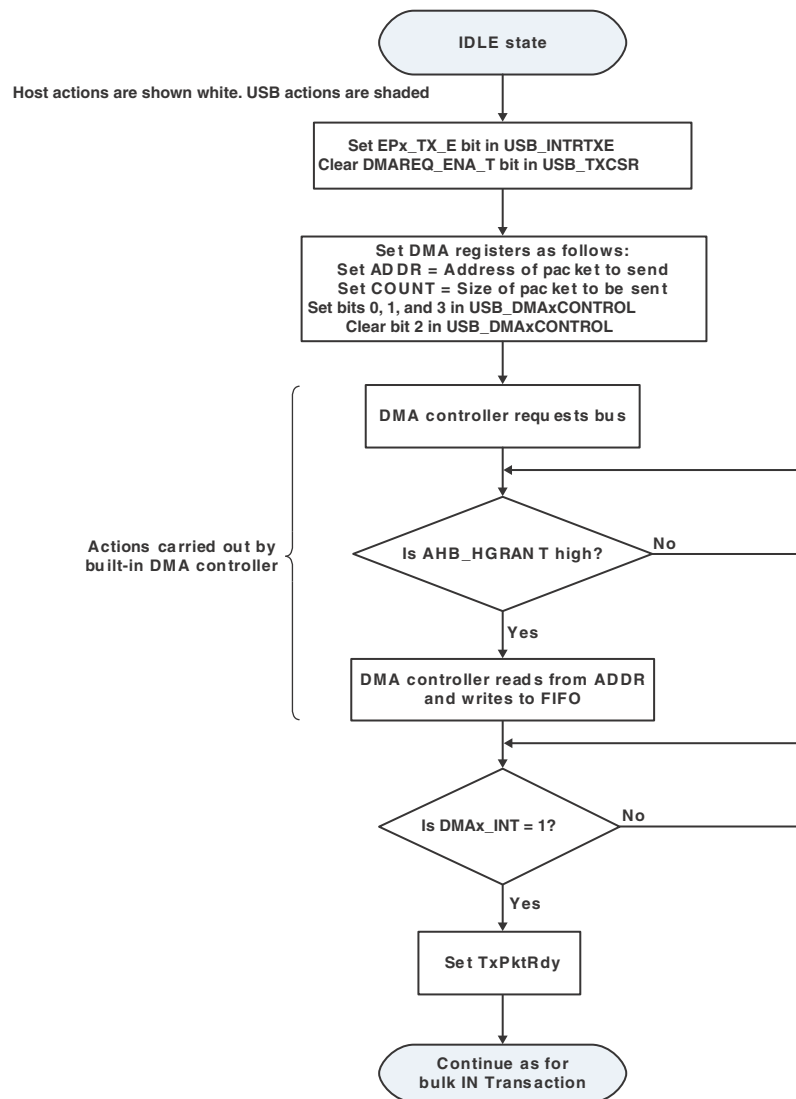


**Figure 22-29:** Single Packet Transmit During DMA Operation

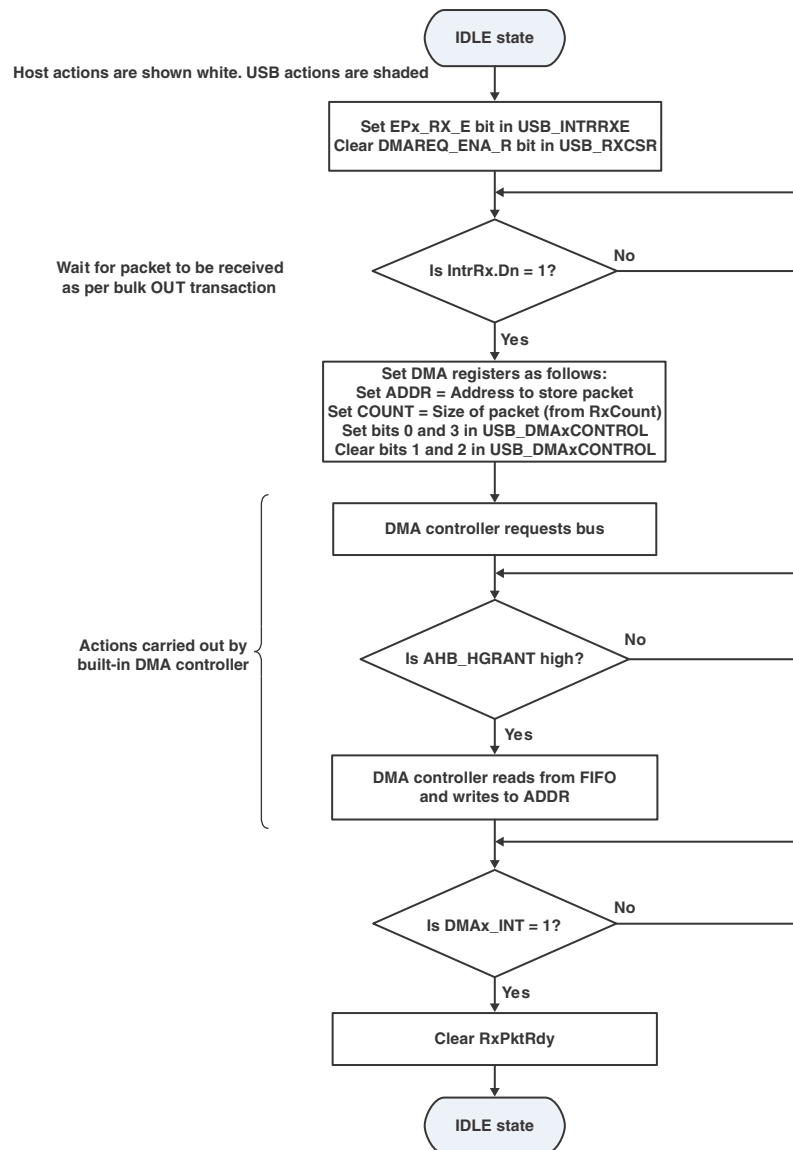


**Figure 22-30:** Single Packet Receive During DMA Operation

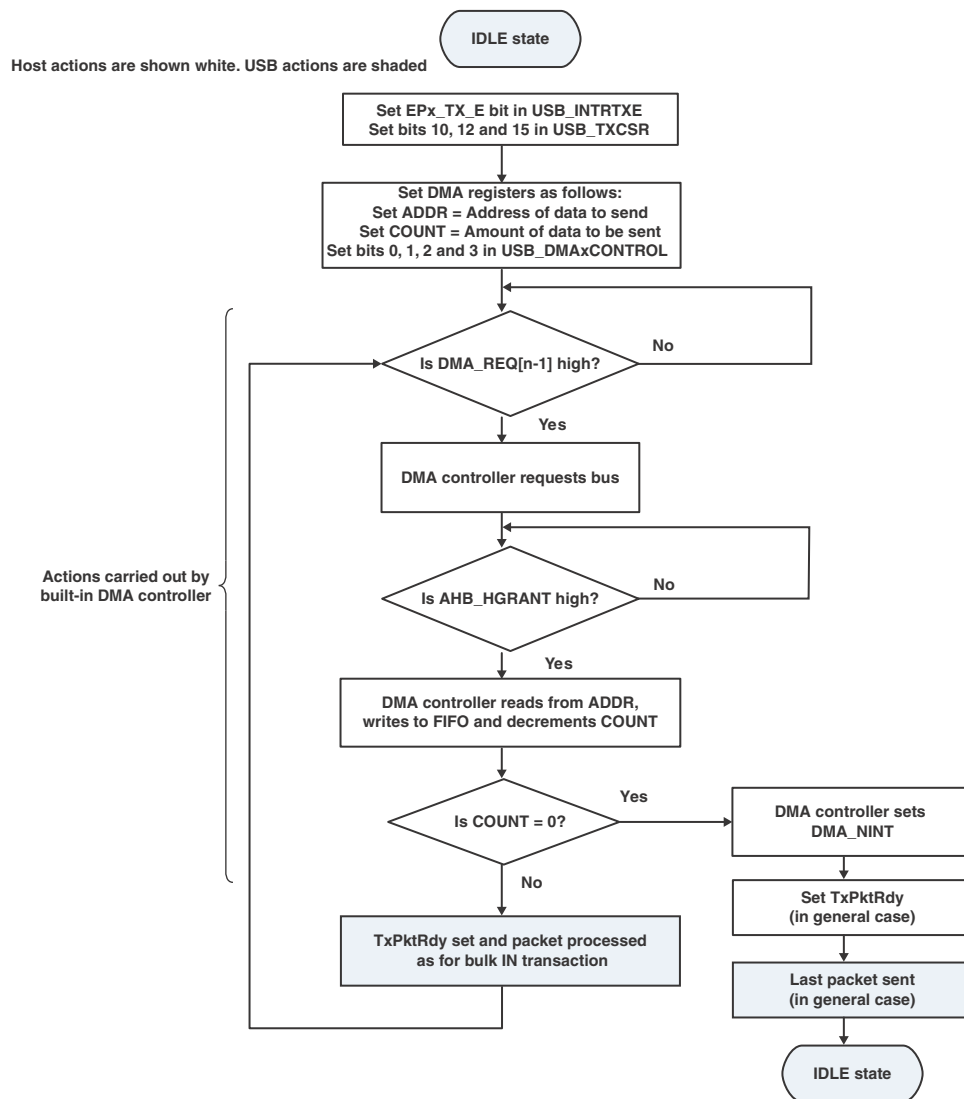


**Figure 22-31:** Multiple Packet Transmit During DMA Operation



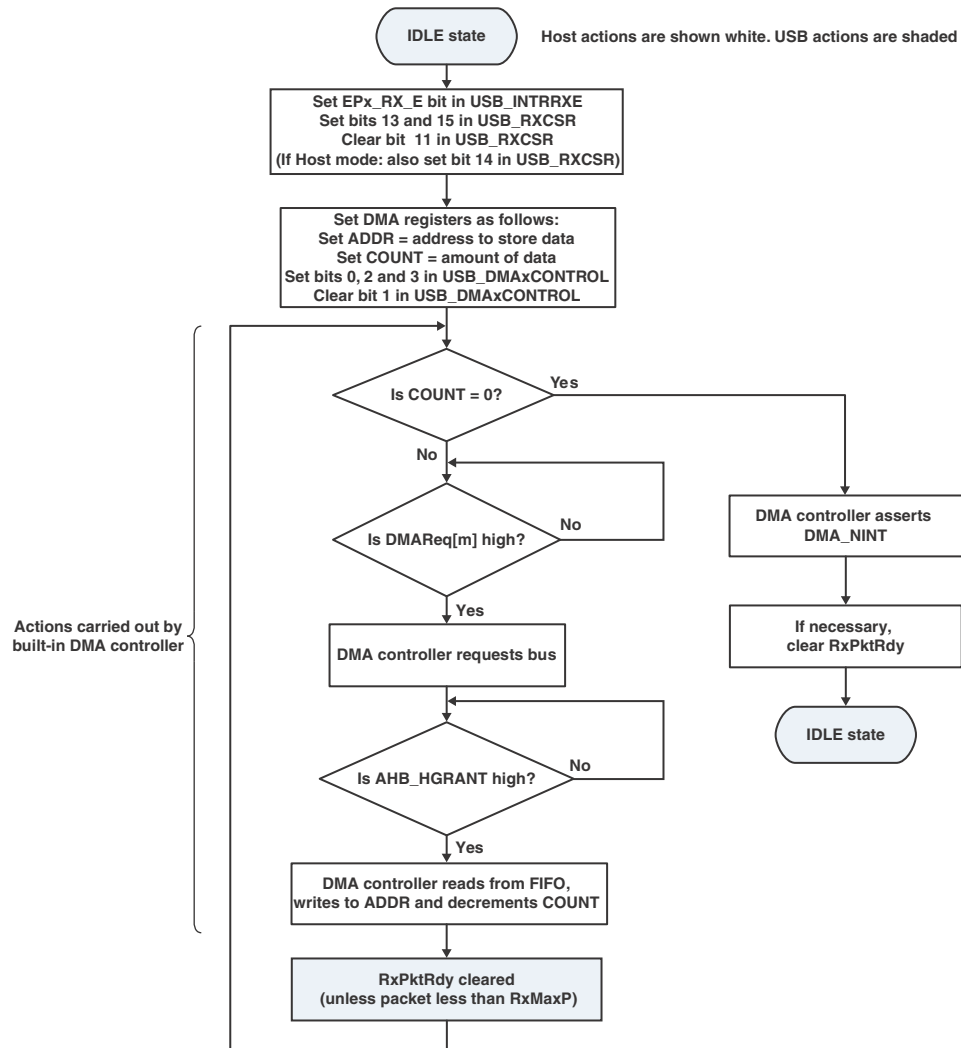


**Figure 22-32:** Multiple Packet Receive During DMA Operation (Data Size Known)

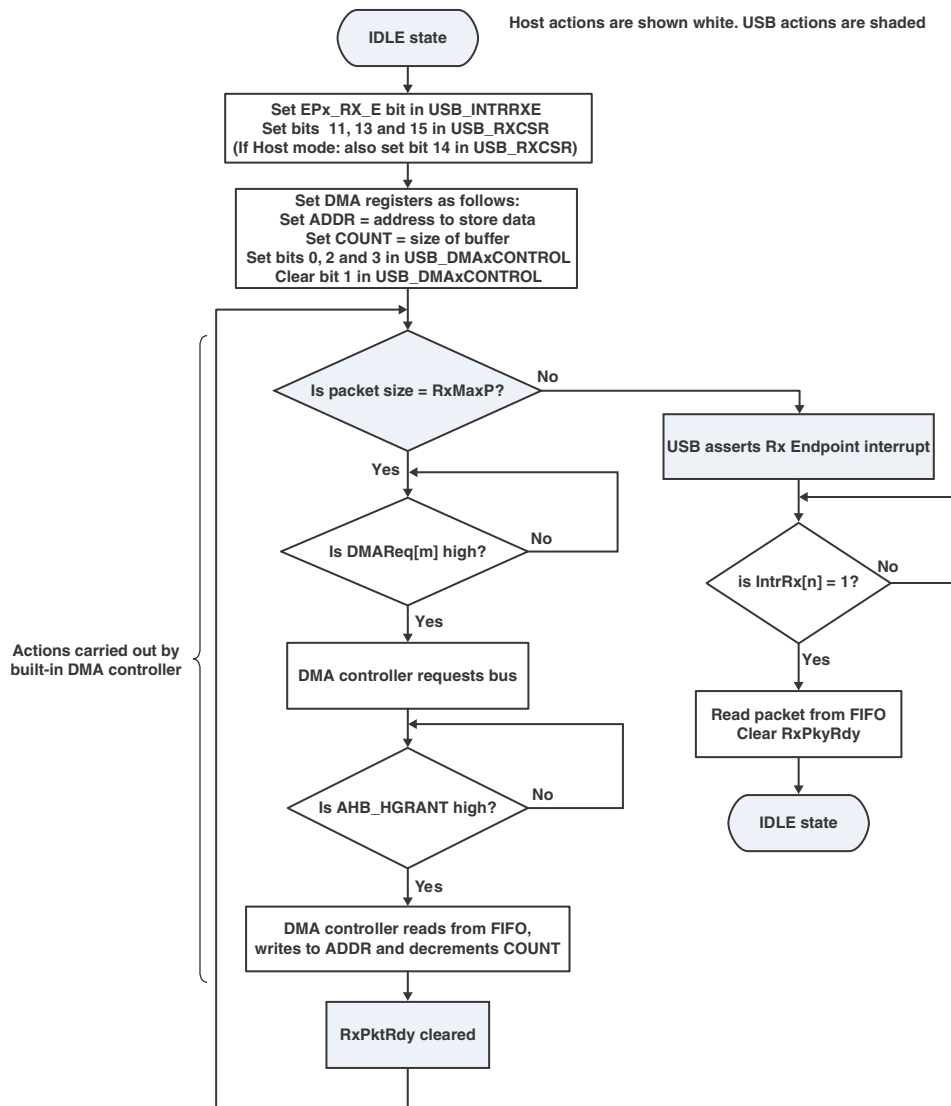


**Figure 22-33:** Multiple Packet Receive During DMA Operation (Data Size Not-known)

## OTG Session Request

In order to conserve power, the USB on-the-go supplement allows VBUS to only be powered up when required and to be turned off when the bus is not in use.

VBUS is always supplied by the A device on the bus. The USB controller determines whether it is the A device or the B device by sampling the `USB_ID` input from the PHY. This signal is pulled low when an A-type plug is sensed (signifying that the USB controller is the A device), but the input is taken high when a B-type plug is sensed (signifying that the USB controller is the B device).

## Starting a Session

When the device containing the USB controller wants to start a session, the processor core must set the `SESSION` bit in the `USB_DEV_CTL` register. The USB controller then enables ID pin sensing. This results in the `USB_ID` input either being taken low if an A-type connection is detected or high if a B-type connection is detected. The `B_DEVICE` bit in the `USB_DEV_CTL` register is also set to indicate whether the USB controller has adopted the role of the A device or the 'B' device.

If the USB controller is the A device: The USB controller then enters host mode (the A device is always the default host), and waits for VBUS to go above the VBUS valid threshold, as indicated when the `VBUS1-0` bits in the `USB_DEV_CTL` register go to 11.

The USB controller then waits for a peripheral to be connected. When a peripheral is detected, a connect interrupt (`CONN_B` bit in `USB_INTRUSB`) is generated (if enabled) and either the `FSDEV` or `LSDEV` bit in the `USB_DEV_CTL` register is set, depending on whether a full-speed peripheral or a low-speed peripheral was detected. The processor core should then reset this peripheral. To end the session, the processor core should clear the `SESSION` bit in `USB_DEV_CTL`.

If the USB controller is the 'B' device: The USB controller requests a session using the session request protocol defined in the USB on-the-go supplement. This is accomplished by setting the `SESSION` bit in the `USB_DEV_CTL` register.

At the end of the session, the `SESSION` bit is cleared - usually by the USB controller but it can also be cleared by the processor core if the application software wishes to perform a software disconnect. For more information, see the description of the `USB_DEV_CTL` register. The USB controller switches on the pull-up resistor on D+. This signals to the A device to end the session.

## Detecting Activity

When the other device of the OTG set-up wants to start a session, it either raises VBUS above the session valid threshold (if it is the 'A' device as indicated by the `VBUS1-0` bits in the `USB_OTG_DEV_CTL` register going to 10), or (if it is the 'B' device) first pulses the data line then pulses VBUS. Depending on which of these actions happens, the USB controller can determine whether it is the 'A' device or the 'B' device in the current set-up and act accordingly.

If VBUS is raised above the session valid threshold, the USB controller is the 'B' device. The USB controller sets the `SESSION` bit in the `USB_OTG_DEV_CTL` register. When reset signaling is detected on the bus, a reset interrupt (`RESET_OR_BABLE_B = 1`) is generated (if enabled) that the processor core should interpret as the start of a session. The USB controller is in peripheral mode at this point as the 'B' device is the default peripheral.

At the end of the session, the 'A' device turns off the power to VBUS. When VBUS drops below the session valid threshold (as indicated by the `VBUS1-0` bits in the `USB_OTG_DEV_CTL` register going to 01), the USB controller detects this and clears the `SESSION` bit to indicate that the session has ended. A disconnect interrupt (`DISCON_B` bit in `USB_INTRUSB`) is also generated (if enabled).

If data line/VBUS pulsing is detected, the USB controller is the 'A' device. The controller generates a `SESSION_REQ_B` interrupt (bit 6 in `USB_INTRUSB`, if enabled) to indicate that the 'B' device is requesting a session. The processor core should then start a session by setting the `SESSION` bit.

## Host Negotiation/Configuration

When the USB controller is the A device (`USB_ID` low, `B_DEVICE`= 0), the controller automatically enters host mode when a session starts.

When the USB controller is the B device (`USB_ID` high, `B_DEVICE`= 1), the controller automatically enters peripheral mode when a session starts. The processor core can request that the USB controller become the host by setting the `HOST_REQ` bit in the `USB_OTG_DEV_CTL` register. This bit can be set either when requesting a session start by setting the `SESSION` bit in `USB_OTG_DEV_CTL` or at any time after a session has started. When the USB controller next enters suspend mode (no activity on the bus for 3 ms), and assuming the `HOST_REQ` bit remains set, the controller enters host mode and begins host negotiation (as specified in the USB OTG supplement), causing the PHY to disconnect the pull-up resistor on the D+ line. This should cause the A device to switch to peripheral mode and to connect its own pull-up resistor. When the USB controller detects this, it generates a connect interrupt (`CONN_B` bit in `USB_INTRUSB`) if this is enabled. The controller also sets the `RESET` bit in the `USB_POWER` register to begin resetting the A device. (The USB controller begins this reset sequence automatically to ensure that reset is started as required within 1 ms of the A device connecting its pull-up resistor). The processor core should wait at least 20 ms, then clear the `RESET` bit and enumerate the A device.

When the USB controller-based B device has finished using the bus, the processor core should put it into suspend mode by setting the `SUSPEND_MODE` bit in the `USB_POWER` register. The A device should detect this and either terminate the session or revert to host mode. If the A device is USB controller-based, it generates a disconnect interrupt (`DISCON_B` bit in `USB_INTRUSB`) if this is enabled.

## Wakeup from Hibernate State

To conserve power when the chip is idle, systems often use power-down modes to shut down power and clocks to various parts of the chip. Hibernate state saves the most power (core clock, peripherals clocks, and internal power are off; only external power is on).

During the course of normal operation, the software can decide that the chip has been idle for a long enough period that there is no immediate need for the clocks to be active and the chip can be put into a power-down mode such as hibernate. This period of inactivity occurs when there is a USB suspend state (idle on the bus for greater than 3 ms) or if no OTG session is valid. The `SUSPEND_MODE` bit (in `USB_POWER`) and `VBUS1-0` status bits (in `USB_OTG_DEV_CTL`) are used to indicate these states.

Before the system software (driver) pushes processor into the hibernate state, the software has to make sure that the `HIBER` bit (in `USB_PHY_CTL`) is set. Setting this bit activates the non-idle activity detection logic in the PHY. Any non-idle activity on the USB bus is detected by the non-idle activity detection logic in the analog PHY. This logic wakes up the processor and generates a low to high transition on `EXT_WAKE` pin.

To be able to use non-idle activity detection logic as a wakeup source for the processor, enable the USB wakeup source by programming the appropriate bits in the voltage regulator control register (VR\_CTL). After the processor wakes up, USB is listed as the wakeup source in the PLL status (PLL\_STAT) register. The EXT\_WAKE pin can be used by the external power-up sequence chip to power up SDRAM or an other external peripheral. The processor typically goes through these steps (see [Timing Diagram of EXT\\_WAKE Pin](#)) when it comes out of hibernate state.

After the chip comes out of hibernate state, the software has to make sure that the RESTORE bit of the USB\_PHY\_CTL register is set. This setting deactivates the non-idle activity detection logic and ensures proper USB functionality.

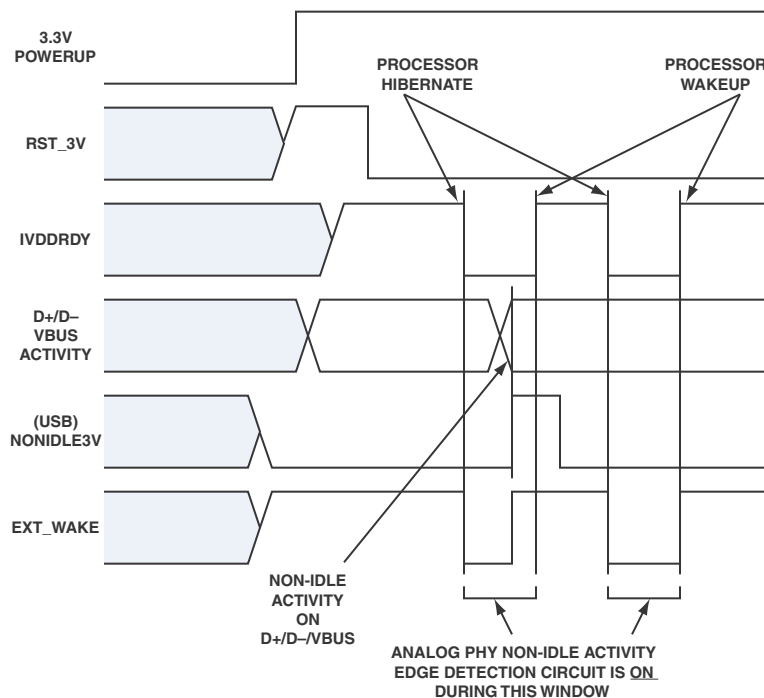


**Figure 22-34:** Timing Diagram of EXT\_WAKE Pin

The interrupt will be asserted when either of the following events occur:

- Non-idle signaling occurs during the USB suspend state (including USB reset signaling)
- VBUS falls below the session valid threshold

## Data Transfer

Regardless of whether the USB controller is operating in host or peripheral mode, data is channeled through the endpoint FIFOs to construct packets to be sent or to be received over the USB. The RX FIFOs are used to receive OUT packets when in peripheral mode and IN packets when operating in host mode.

Similarly, the TX FIFOs are used to transmit IN packets when in peripheral mode and OUT packets as a host.

Data may be moved between the FIFOs and memory using either DMA or core accesses. Each endpoint FIFO has its own individually programmable options so that each can be set up separately. Different transfer types must be treated differently by the system. Data transfers of significant size almost certainly require DMA to move the data around; but smaller packet sizes might be handled completely by the processor.

Each data endpoint supports both double and single-buffering modes. In single-buffered operation, FIFOs are unloaded and loaded on a packet-by-packet basis. Double-buffering imposes less burden on the system by allowing two packets to be buffered in a FIFO before it is necessary to use DMA/interrupts to service the FIFO. Double-buffering mode is automatically enabled when a *MaxPktSize* is set for an endpoint that is equal to or less than half the size in bytes of that FIFO.

## Loading/Unloading Packets from Endpoints

Transfers to and from the FIFOs can be 32-bit, 16-bit, or 8-bit as required. When using core accesses the same width must be used for transfers associated with one data packet, so that data is consistently byte, half-word or word aligned. The last transfer may, however, contain fewer bytes than the previous transfers in order to complete an 8-bit or 16-bit transfer.

When using the DMA to access the FIFOs, the only requirement is that the starting DMA address be word aligned, or aligned on a 32-bit boundary. The packet transfer starts with a word transfer, but half-word and/or byte transfers may be added at the end to handle any residue.

## DMA Master Channels

The USB controller provides eight DMA master channels to provide a more efficient transfer of larger amounts of data between the FIFOs and the processor core; and to free up the processor core for other tasks. Each of these channels is configured and controlled using the DMA control registers.

Each DMA controller can operate in one of two DMA modes: 0 or 1. When operating in mode 0, the DMA controller only can be programmed to load or unload one packet, so processor intervention is required for each packet transferred over the USB. This mode can be used with any endpoint, whether it uses control, bulk, isochronous, or interrupt transactions.

When operating in DMA mode 1, the DMA controller can only be programmed to load/unload a complete bulk transfer, which can be many packets. After set up, the DMA controller loads or unloads the packets, interrupting the processor only when the transfer has completed. DMA mode 1 can only be used with endpoints that use bulk transactions. DMA mode 1 is most valuable where large blocks of data are transferred to a bulk endpoint. The USB protocol requires such packets to be split into a series of packets of *MaxPktSize* for the endpoint. Mode 1 can be used to avoid the overhead of having to interrupt the processor after each individual packet; instead the processor is only interrupted after the transfer has completed. In some cases, the block of data transferred comprises a predefined number of these packets

that the controlling software counts through the transfer process. In other cases, the last packet in the series may be less than the maximum packet size and the receiver may use this "short" packet to signal the end of the transfer. If the total size of the transfer is an exact multiple of the maximum packet size, the transmitting software should send a null packet for the receiver to detect.

Each channel can be independently programmed for the selected operating mode.

For bulk OUT transfers using DMA mode 1, the DMA request line is asserted only when there is an edge transition of the state of the `RXPCKTRDY` and a payload of `MaxPacketSize` has been received. If a data packet has been sitting in the FIFO prior to setting `DMAREQMODE1` in `USB_RXCSR`, the DMA request line will not be asserted when the DMA is enabled in the `DMAx_Control` register. This will cause the data not to be read from the RX FIFO, resulting in a DMA "hang". However, since the packet arrived before `DMAREQMODE` and `DMAREQ_ENA` were enabled in `USB_RXCSR`, an RX interrupt will be generated for the corresponding endpoint. Therefore, the software should set the `DMAREQMODE` to Request Mode 0 to unload the pre received packet. The RX interrupt service routine may be similar to the following.

```
If USB_RXCOUNT == MaxPktSize
```

```
    Switch to DMA Mode 0 and unload the packet (in Mode 0, DMA_REQ is always asserted whenever there is data in the FIFO)
```

```
    You should set the DMA_COUNT to MaxPktSize so as to unload only one packet
```

```
    If AUTOCLEAR is set, you do not need to manually clear RXPCKTRDY
```

```
    Switch back to DMA Mode 1 and set the count to
```

```
    (Total_Count - MaxPktSize)
```

```
Else
```

```
    Handle as normal for case of short packet
```

DMA transfers may be 8-bit, 16-bit, or 32-bit. All the transfer associated with one packet (with the exception of the last) must be of the same width, so that the data is consistently byte-aligned or word-aligned. The last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

## DMA Bus Cycles

The DMA controller uses incrementing bursts of an unspecified length on the peripheral DMA bus. The controller starts a new burst when it is first granted bus mastership (whether at the start of a USB packet or when regaining the bus after being thrown off part way through a packet) and when the peripheral address starts a new 1K byte block.

When unloading packets from the FIFOs, the DMA controller requests ahead to the USB controller. Although it starts the transfer with two BUSY cycles while it is getting the first word from the FIFO, all subsequent words of the packet are immediately available. No further BUSY cycles are required. The DMA

controller is associated with a two-word buffer, so no data is lost if it loses bus mastership part way through unloading a packet. When bus mastership is regained, it can continue unloading the packet without adding any BUSY cycles.

The DMA start address (written to the `DMAxADDR`) must be word aligned.

Split transactions and retries are supported.

The DMA request lines are individually enabled through the `DMAReqEnab` bit in the appropriate `TxCSR` control register or `RxCSR` control register and operate in two modes, referred to as DMA Request Mode 0 and DMA Request Mode 1. The choice of operating mode is made through the `DMAReqMode` bit in the `TxCSR` for Tx endpoints or `RxCSR` for Rx endpoints).

**NOTE:** When operating in Host mode, if either the `RxStall` bit (bit 5 of `TxCSRL` register) or the Error bit (bit 2 of `TxCSRL` register) becomes set following three failed attempts to transmit a packet, the DMA request line will be disabled until the `RxStall/Error` bit(s) have been cleared.

The mode selected also affects the generation of Endpoint interrupts (if enabled). In DMA Request Mode 0, no interrupt is generated when packets are received but the appropriate Endpoint interrupt is generated to prompt the loading of all packets. In DMA Request Mode 1, the Endpoint interrupt is suppressed except following the receipt of a short packet (one of less than `RxMaxP` bytes).

**Table 22-7:** Endpoint Interrupt associated with `RxPktRdy` being set

| <code>DMAReqEnab</code> | <code>DMAReqMode</code> | EP Interrupt Generated? |
|-------------------------|-------------------------|-------------------------|
| 0                       | X                       | YES                     |
| 1                       | 0                       | NO                      |
| 1                       | 1                       | Only is short packet    |

**Table 22-8:** Endpoint Interrupt associated with `RxPktRdy` being set

| <code>DMAReqEnab</code> | <code>DMAReqMode</code> | EP Interrupt Generated? |
|-------------------------|-------------------------|-------------------------|
| 0                       | X                       | YES                     |
| 1                       | 0                       | YES                     |
| 1                       | 1                       | NO                      |

**NOTE:** `Tx/RxMaxP` must be set to an even number of bytes for proper interrupt generation in DMA Mode 1.

DMA transfers may be 8-bit, 16-bit, or 32-bit as required. However, all the transfers associated with one packet (with the exception of the last) must be of the same width so that the data is consistently byte-,



word- or double-word-aligned. The last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

**NOTE:** DMA Requests should be disabled before the DMA Request Mode is changed. In particular, the DMAReqMode bit in the TxCSRH register should not be set to zero either before or in the same cycle as the corresponding DMAReqEnab bit is cleared to zero.

## Transferring Packets Using DMA

Use of the DMA master channels to access the USB controller FIFOs requires that both the appropriate channel and the endpoint be programmed appropriately. Many variations are possible. The following sections detail the standard setups used for the basic actions of transferring individual packets and multiple packets.

### Individual Packet: RX Endpoint

The transfer of individual packets is normally carried out using DMA mode 0. The USB controller RX endpoint is programmed as follows:

1. The relevant EPx\_RX\_E bit in the USB\_INTRRXE register is set to 1.
2. The DMA\_ENA bit of the appropriate USB\_RXCSR register is set to 0. (There is no need to set the USB controller to support DMA for this operation.)
3. When a packet is received by the USB controller, it generates the appropriate endpoint interrupt (using USB\_INTRRX). The processor should then program the appropriate DMA master channel as follows:
  - DMAxADDR: memory address to store packet
  - USB\_DMAxCOUNT: size of packet (determined by reading the USB controller USB\_RXCOUNT register)
  - USB\_DMAxCONTROL: INT\_ENA = 1, DMA\_ENA = 1, DIRECTION = 0, DMAREQMODE\_R = 0

The DMA controller then requests bus mastership and transfers the packet to memory. It interrupts the processor when it has completed the transfer. The processor should then clear the RXPkTRDY bit in the USB\_RXCSR register.

### Individual Packet: TX Endpoint

Again using DMA mode 0, a USB controller TX endpoint is programmed as follows.

1. The relevant EPx\_TX\_E bit in the USB\_INTRTXE register is set to 1.
2. The DMA\_ENA bit of the appropriate USB\_TxCSR register is set to 0. (There is no need to set the USB controller to support DMA for this operation.)

3. When the FIFO can accommodate data, the USB controller interrupts the processor with the appropriate TX endpoint interrupt. The processor should then program the DMA channel as follows:
  - `DMAxADDR`: memory address of packet to send
  - `USB_DMAxCOUNT`: size of packet to be sent
  - `USB_DMAxCONTROL`: `INT_ENA = 1`, `DMA_ENA = 1`, `DIRECTION = 1`, `DMAREQMODE_T = 0`

The DMA controller then requests bus mastership and transfers the packet to the USB controller FIFO. When it has completed the transfer, it generates a DMA interrupt. The processor should then set the `TXPK-TRDY` bit in the `USB_TXCSR` register.

## Multiple Packets: RX Endpoint

Multiple packets normally are transferred using DMA mode 1. The DMA controller is programmed using the DMA registers:

- `DMAxADDR`: memory address of the buffer in which to store transfer
- `USB_DMAxCOUNT`: maximum size of data buffer
- `USB_DMAxCONTROL`: `INT_ENA = 1`, `DMA_ENA = 1`, `DIRECTION = 0`, `DMAREQMODE_R = 1`

The USB controller RX endpoint should now be programmed as follows:

1. The relevant `EPx_RX_E` bit in the `USB_INTRRXE` register is set to 1.
2. The `AUTOCLEAR_R`, `DMAREQ_ENA_R` and `DMAREQMODE_R` bits of the appropriate `USB_RXCSR` register is set to 1. In host mode, the `AUTOREQ_RH` and `DMAREQMODE_RH` bits should also be set to 1.

As each packet is received by the USB controller, the DMA master channel requests bus mastership and transfers the packet to memory. With `AUTOCLEAR_R` set, the USB controller automatically clears its `RXPK-TRDY` bit. This process continues automatically until the USB controller receives a short packet (one of less than the maximum packet size for the endpoint) signifying the end of the transfer. This short packet is not transferred by the DMA controller: instead the USB controller interrupts the processor by generating the appropriate endpoint interrupt. The processor can then read the `USB_RXCOUNT` register to see the size of the short packet and either unload it manually or reprogram the DMA controller in mode 0 to unload the packet.

The `DMAxADDR` register is incremented as the packets are unloaded, so the processor can determine the size of the transfer by comparing the current value of `DMAxADDR` with the start address of the memory buffer.

If the size of the transfer exceeds the data buffer size, the DMA controller stops unloading the FIFO and interrupts the processor.

## Multiple Packets: TX Endpoints

Using DMA mode 1 for a TX endpoint, the DMA controller is programmed as follows:

- **DMAxADDR:** memory address of data block to send
- **USB\_DMAxCOUNT:** size of data block
- **USB\_DMAxCONTROL:** `INT_ENA = 1`, `DMA_ENA = 1`, `DIRECTION = 1`, `DMAREQMODE_T = 1`

The USB controller TX endpoint is programmed as follows:

1. The relevant `EPx_TX_E` bit in the `USB_INTRTXE` register is set to 1.
2. The `AUTOSET_T` and `DMA_ENA` bits of the appropriate `USB_EP_NIx_TXCSR` register is set to 1.

When the FIFO in the USB controller becomes available, the DMA controller requests bus mastership and transfers a packet to the FIFO. With `AUTOSET_T` set, the USB controller automatically sets the `TXPKTRDY` bit. This process continues until the entire data block is transferred to the USB controller. The DMA controller then interrupts the processor by taking `DMAx_INT` low. If the last packet to be loaded was less than the maximum packet size for the endpoint, the `TXPKTRDY` bit is not set for this packet; the processor should respond to the DMA interrupt by setting the `TXPKTRDY` bit to allow the last short packet to be sent. If the last packet to be loaded was of the maximum packet size, then the action to take depends on whether the transfer is under the control of an application such as the mass storage software on Windows system that keeps count of the individual packets sent. If the transfer is not under such control, the processor should still respond to the DMA interrupt by setting the `TXPKTRDY` bit. This has the effect of sending a null packet for the receiving software to interpret as indicating the end of the transfer.

## USB Register Descriptions

Universal Serial Bus Controller (USB) contains the following registers.

**Table 22-9:** USB Module Registers

| Name        | Description                        |
|-------------|------------------------------------|
| USB_FADDR   | Function Address Register          |
| USB_POWER   | Power and Device Control Register  |
| USB_INTRTX  | Transmit Interrupt Register        |
| USB_INTRRX  | Receive Interrupt Register         |
| USB_INTRTXE | Transmit Interrupt Enable Register |
| USB_INTRRXE | Receive Interrupt Enable Register  |

**Table 22-9:** USB Module Registers (Continued)

| Name               | Description                            |
|--------------------|----------------------------------------|
| USB_IRQ            | Common Interrupts Register             |
| USB_IEN            | Common Interrupts Enable Register      |
| USB_FRAME          | Frame Number Register                  |
| USB_INDEX          | Index Register                         |
| USB_TESTMODE       | Testmode Register                      |
| USB_FIFOBn         | FIFO Byte (8-Bit) Register             |
| USB_FIFOHn         | FIFO Half-Word (16-Bit) Register       |
| USB_FIFOn          | FIFO Word (32-Bit) Register            |
| USB_DEV_CTL        | Device Control Register                |
| USB_TXFIFOSZ       | Transmit FIFO Size Register            |
| USB_RXFIFOSZ       | Receive FIFO Size Register             |
| USB_TXFIFOADDR     | Transmit FIFO Address Register         |
| USB_RXFIFOADDR     | Receive FIFO Address Register          |
| USB_EPINFO         | Endpoint Information Register          |
| USB_RAMINFO        | RAM Information Register               |
| USB_LINKINFO       | Link Information Register              |
| USB_VPLEN          | VBUS Pulse Length Register             |
| USB_HS_EOF1        | High-Speed EOF 1 Register              |
| USB_FS_EOF1        | Full-Speed EOF 1 Register              |
| USB_LS_EOF1        | Low-Speed EOF 1 Register               |
| USB_SOFT_RST       | Software Reset Register                |
| USB_MPN_TXFUNCADDR | MPn Transmit Function Address Register |

**Table 22-9:** USB Module Registers (Continued)

| Name               | Description                                                 |
|--------------------|-------------------------------------------------------------|
| USB_MPn_TXHUBADDR  | MPn Transmit Hub Address Register                           |
| USB_MPn_TXHUBPORT  | MPn Transmit Hub Port Register                              |
| USB_MPn_RXFUNCADDR | MPn Receive Function Address Register                       |
| USB_MPn_RXHUBADDR  | MPn Receive Hub Address Register                            |
| USB_MPn_RXHUBPORT  | MPn Receive Hub Port Register                               |
| USB_EPn_TXMAXP     | EPn Transmit Maximum Packet Length Register                 |
| USB_EP0_CSRn_H     | EP0 Configuration and Status (Host) Register                |
| USB_EP0_CSRn_P     | EP0 Configuration and Status (Peripheral) Register          |
| USB_EPn_TXCSR_H    | EPn Transmit Configuration and Status (Host) Register       |
| USB_EPn_TXCSR_P    | EPn Transmit Configuration and Status (Peripheral) Register |
| USB_EPn_RXMAXP     | EPn Receive Maximum Packet Length Register                  |
| USB_EPn_RXCSR_H    | EPn Receive Configuration and Status (Host) Register        |
| USB_EPn_RXCSR_P    | EPn Receive Configuration and Status (Peripheral) Register  |
| USB_EP0_CNTn       | EP0 Number of Received Bytes Register                       |
| USB_EPn_RXCNT      | EPn Number of Bytes Received Register                       |
| USB_EPn_TXTYPE     | EPn Transmit Type Register                                  |
| USB_EP0_TYPERn     | EP0 Connection Type Register                                |
| USB_EP0_NAKLIMITn  | EP0 NAK Limit Register                                      |
| USB_EPn_TXINTERVAL | EPn Transmit Polling Interval Register                      |
| USB_EPn_RXTYPE     | EPn Receive Type Register                                   |
| USB_EPn_RXINTERVAL | EPn Receive Polling Interval Register                       |
| USB_EP0_CFGDATAn   | EP0 Configuration Information Register                      |

**Table 22-9:** USB Module Registers (Continued)

| Name                       | Description                                   |
|----------------------------|-----------------------------------------------|
| USB_DMA_IRQ                | DMA Interrupt Register                        |
| USB_DMA <sub>n</sub> _CTL  | DMA Channel n Control Register                |
| USB_DMA <sub>n</sub> _ADDR | DMA Channel n Address Register                |
| USB_DMA <sub>n</sub> _CNT  | DMA Channel n Count Register                  |
| USB_RQPKTCNT <sub>n</sub>  | EP <sub>n</sub> Request Packet Count Register |
| USB_CT_UCH                 | Chirp Timeout Register                        |
| USB_CT_HHSRTN              | Host High Speed Return to Normal Register     |
| USB_CT_HSBT                | High Speed Timeout Register                   |
| USB_LPM_ATTR               | LPM Attribute Register                        |
| USB_LPM_CTL                | LPM Control Register                          |
| USB_LPM_IEN                | LPM Interrupt Enable Register                 |
| USB_LPM_IRQ                | LPM Interrupt Status Register                 |
| USB_LPM_FADDR              | LPM Function Address Register                 |
| USB_VBUS_CTL               | VBUS Control Register                         |
| USB_BAT_CHG                | Battery Charging Control Register             |
| USB_PHY_CTL                | PHY Control Register                          |
| USB_PLL_OSC                | PLL and Oscillator Control Register           |

## Function Address Register

The USB\_FADDR register contains the device address used in peripheral mode. The processor writes this register with the address received through a SET\_ADDRESS command from the host.

USB\_FADDR: Function Address Register - R/W

Reset = 0x00

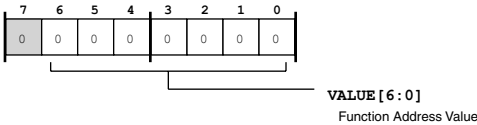


Figure 22-35: USB\_FADDR Register Diagram

Table 22-10: USB\_FADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Function Address Value.<br>The USB_FADDR.VALUE bits contain the address of the peripheral part of the transaction. |

Power and Device Control Register

The USB\_POWER register controls suspend and resume signaling and controls some operational aspects of the USB controller.

USB\_POWER: Power and Device Control Register - R/W

Reset = 0x20

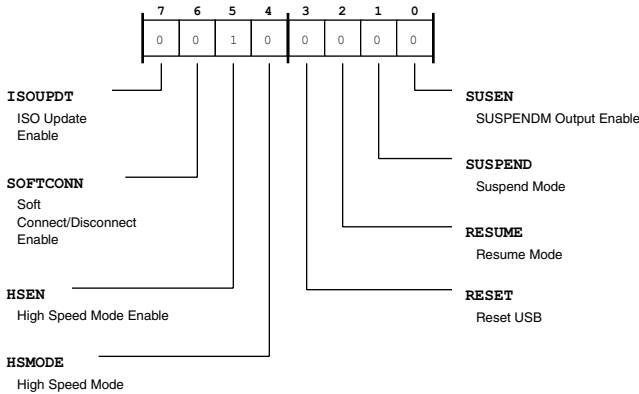


Figure 22-36: USB\_POWER Register Diagram

**Table 22-11:** USB\_POWER Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | ISOUPDT  | ISO Update Enable.<br>The <code>USB_POWER.ISOUPDT</code> bit directs the USB controller to wait for an SOF token from the time <code>TXPKTRDY</code> is set before sending the packet. If an IN token is received before an SOF token, the USB controller sends a zero length data packet. This <code>USB_POWER.ISOUPDT</code> bit only affects endpoints performing isochronous transfers. This bit only is valid in peripheral mode ( <code>USB_DEV_CTL.HOSTMODE = 0</code> ).                                                                                                                                             |
|         |          | 0      Disable ISO Update                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1      Enable ISO Update                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 6       | SOFTCONN | Soft Connect/Disconnect Enable.<br>In peripheral mode, the D+/- lines default to disconnected. Setting this bit will enable the D+/- termination resistors. This bit is automatically set when the <code>DevCtl.Session</code> bit is written with '1'. The <code>USB_POWER.SOFTCONN</code> bit enables USB controller soft connect/disconnect, enabling the termination resistors for <code>USB_DP</code> (Data +) and <code>USB_DM</code> (Data -) pins. When disabled, these pins are three-stated. Note that <code>USB_POWER.SOFTCONN</code> only is valid in peripheral mode ( <code>USB_DEV_CTL.HOSTMODE = 0</code> ). |
|         |          | 0      Disable Soft Connect/Disconnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable Soft Connect/Disconnect                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 5       | HSEN     | High Speed Mode Enable.<br>The <code>USB_POWER.HSEN</code> bit enables USB controller negotiation for high speed when the device is reset by the hub/host. If disabled, the USB controller only operates in full-speed mode.                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 0      Disable Negotiation for HS Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable Negotiation for HS Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



**Table 22-11:** USB\_POWER Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | HSMODE   | High Speed Mode.<br>The USB_POWER.HSMODE bit indicates whether or not the USB controller successfully negotiated high-speed mode during a USB controller reset. In peripheral mode (USB_DEV_CTL.HOSTMODE = 0), this bit has valid data when the USB controller completes reset. In host mode (USB_DEV_CTL.HOSTMODE = 1), this bit has valid data when the USB_IRQ.RSTBABBLE bit is cleared, remaining valid for the duration of the session.                                                  |
|         |          | 0 Full Speed Mode (HS fail during reset)                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 High Speed Mode (HS success during reset)                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 3       | RESET    | Reset USB.<br>The USB_POWER.RESET bit indicates (in both host and peripheral modes) that the USB controller has detected that reset signaling is present on the bus. In peripheral mode (USB_DEV_CTL.HOSTMODE = 0), this bit is read only, but in host mode (USB_DEV_CTL.HOSTMODE = 1), this bit is read/write, permitting the processor core to set the bit and initiate a USB controller reset.                                                                                             |
|         |          | 0 No Reset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Reset USB                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 2       | RESUME   | Resume Mode.<br>The USB_POWER.RESUME bit directs the USB controller to generate resume signaling when the function is in suspend mode (USB_POWER.SUSPEND = 1). The processor core should clear this bit after 10 ms (a maximum of 15 ms) to end resume signaling. When the USB controller is in host mode (USB_DEV_CTL.HOSTMODE = 1), the USB controller automatically sets the USB_POWER.RESUME bit when resume signaling from the target is detected while the USB controller is suspended. |
|         |          | 0 Disable Resume Signaling                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Enable Resume Signaling                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

**Table 22-11:** USB\_POWER Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | SUSPEND  | Suspend Mode.<br>When the USB controller is in host mode (USB_DEV_CTL.HOSTMODE = 1), the USB_POWER.SUSPEND bit enables suspend mode. When the USB controller is in peripheral mode (USB_DEV_CTL.HOSTMODE = 0), the USB controller sets the USB_POWER.SUSPEND bit on entry to suspend mode and clears the bit when the processor reads the USB_IRQ register. Note that the USB controller automatically clears this bit if the USB_POWER.RESUME bit is set. |
|         |          | 0      Disable Suspend Mode (Host)                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1      Enable Suspend Mode (Host)                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 0       | SUSEN    | SUSPENDM Output Enable.<br>The USB_POWER.SUSEN bit enables the SUSPENDM output (internal USB controller signal). When enabled, the SUSPENDM output signal is used by the USB controller PHY to power-down its drivers when the USB controller is not active.                                                                                                                                                                                               |
|         |          | 0      Disable SUSPENDM Output                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1      Enable SUSPENDM Output                                                                                                                                                                                                                                                                                                                                                                                                                              |

## Transmit Interrupt Register

The USB\_INTRTX register indicates which interrupts are currently active for endpoint 0 and the transmit (Tx) endpoints. Note that the USB controller automatically clears this register when it is read.

**USB\_INTRTX: Transmit Interrupt Register - R/NW**

Reset = 0x0000

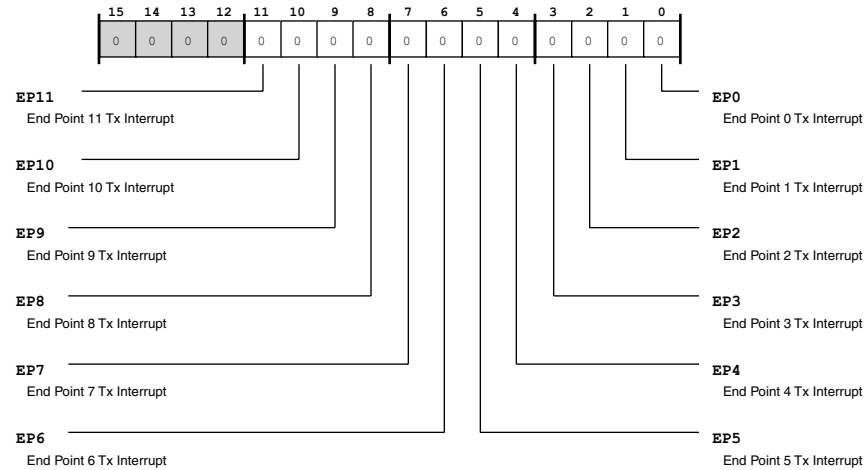


**Figure 22-37:** USB\_INTRTX Register Diagram

**Table 22-12:** USB\_INTRTX Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| 11      | EP11     | End Point 11 Tx Interrupt.<br>The USB_INTRTX.EP11 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                               |
| 10      | EP10     | End Point 10 Tx Interrupt.<br>The USB_INTRTX.EP10 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                               |
| 9       | EP9      | End Point 9 Tx Interrupt.<br>The USB_INTRTX.EP9 bit indicates whether or not a transmit interrupt is pending for this endpoint.   |
|         |          | 0 No Interrupt                                                                                                                    |
|         |          | 1 Interrupt Pending                                                                                                               |

**Table 22-12:** USB\_INTRTX Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| 8       | EP8      | End Point 8 Tx Interrupt.<br>The USB_INTRTX.EP8 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                  |
|         |          | 1 Interrupt Pending                                                                                                             |
| 7       | EP7      | End Point 7 Tx Interrupt.<br>The USB_INTRTX.EP7 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                  |
|         |          | 1 Interrupt Pending                                                                                                             |
| 6       | EP6      | End Point 6 Tx Interrupt.<br>The USB_INTRTX.EP6 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                  |
|         |          | 1 Interrupt Pending                                                                                                             |
| 5       | EP5      | End Point 5 Tx Interrupt.<br>The USB_INTRTX.EP5 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                  |
|         |          | 1 Interrupt Pending                                                                                                             |
| 4       | EP4      | End Point 4 Tx Interrupt.<br>The USB_INTRTX.EP4 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0 No Interrupt                                                                                                                  |
|         |          | 1 Interrupt Pending                                                                                                             |

**Table 22-12:** USB\_INTRTX Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| 3       | EP3      | End Point 3 Tx Interrupt.<br>The USB_INTRTX.EP3 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                             |
|         |          | 1                      Interrupt Pending                                                                                        |
| 2       | EP2      | End Point 2 Tx Interrupt.<br>The USB_INTRTX.EP2 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                             |
|         |          | 1                      Interrupt Pending                                                                                        |
| 1       | EP1      | End Point 1 Tx Interrupt.<br>The USB_INTRTX.EP1 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                             |
|         |          | 1                      Interrupt Pending                                                                                        |
| 0       | EP0      | End Point 0 Tx Interrupt.<br>The USB_INTRTX.EP0 bit indicates whether or not a transmit interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                             |
|         |          | 1                      Interrupt Pending                                                                                        |

## Receive Interrupt Register

The USB\_INTRRX register indicates which interrupts are currently active for the receive (Rx) endpoints. Note that the USB controller automatically clears this register when it is read.

USB\_INTRRX: Receive Interrupt Register - R/NW

Reset = 0x0000

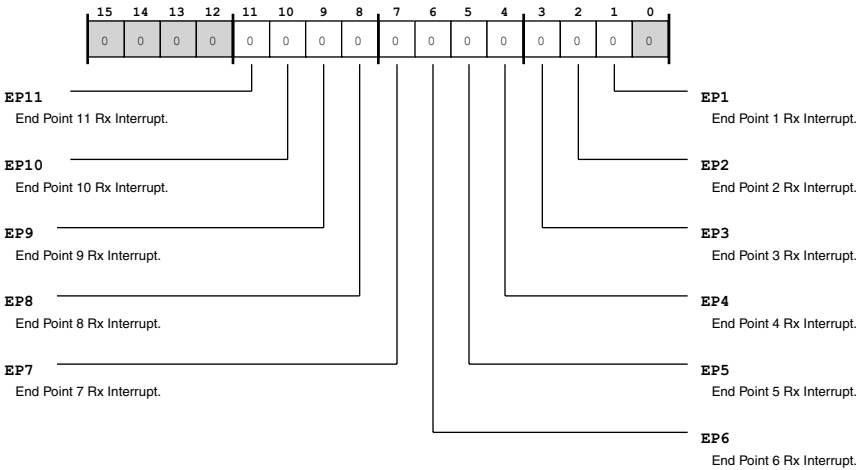


Figure 22-38: USB\_INTRRX Register Diagram

Table 22-13: USB\_INTRRX Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| 11      | EP11     | End Point 11 Rx Interrupt..<br>The USB_INTRRX . EP11 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0No Interrupt                                                                                                                       |
|         |          | 1Interrupt Pending                                                                                                                  |
| 10      | EP10     | End Point 10 Rx Interrupt..<br>The USB_INTRRX . EP10 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0No Interrupt                                                                                                                       |
|         |          | 1Interrupt Pending                                                                                                                  |
| 9       | EP9      | End Point 9 Rx Interrupt..<br>The USB_INTRRX . EP9 bit indicates whether or not a receive interrupt is pending for this endpoint.   |
|         |          | 0No Interrupt                                                                                                                       |
|         |          | 1Interrupt Pending                                                                                                                  |

**Table 22-13:** USB\_INTRRX Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| 8       | EP8      | End Point 8 Rx Interrupt..<br>The USB_INTRRX . EP8 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 7       | EP7      | End Point 7 Rx Interrupt..<br>The USB_INTRRX . EP7 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 6       | EP6      | End Point 6 Rx Interrupt..<br>The USB_INTRRX . EP6 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 5       | EP5      | End Point 5 Rx Interrupt..<br>The USB_INTRRX . EP5 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 4       | EP4      | End Point 4 Rx Interrupt..<br>The USB_INTRRX . EP4 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |

**Table 22-13:** USB\_INTRRX Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------|
| 3       | EP3      | End Point 3 Rx Interrupt..<br>The USB_INTRRX . EP3 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 2       | EP2      | End Point 2 Rx Interrupt..<br>The USB_INTRRX . EP2 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |
| 1       | EP1      | End Point 1 Rx Interrupt..<br>The USB_INTRRX . EP1 bit indicates whether or not a receive interrupt is pending for this endpoint. |
|         |          | 0                      No Interrupt                                                                                               |
|         |          | 1                      Interrupt Pending                                                                                          |

## Transmit Interrupt Enable Register

The USB\_INTRTXE register enables interrupts for endpoint 0 and the transmit (Tx) endpoints. Enabling an interrupt in this register directs the USB controller to generate an interrupt if the corresponding interrupt pending bit in the USB\_INTRTX register is set.



**USB\_INTRTXE: Transmit Interrupt Enable Register - R/W**

Reset = 0x0fff

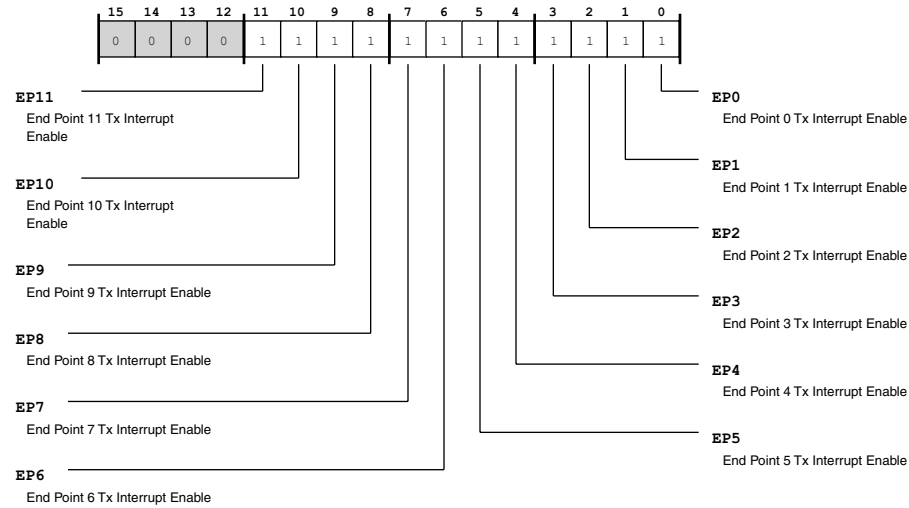


**Figure 22-39:** USB\_INTRTXE Register Diagram

**Table 22-14:** USB\_INTRTXE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                         |                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------|-------------------|
| 11      | EP11     | End Point 11 Tx Interrupt Enable.<br>The USB_INTRTXE.EP11 bit enables the transmit interrupt for this endpoint. |                   |
|         |          | 0                                                                                                               | Disable Interrupt |
|         |          | 1                                                                                                               | Enable Interrupt  |
| 10      | EP10     | End Point 10 Tx Interrupt Enable.<br>The USB_INTRTXE.EP10 bit enables the transmit interrupt for this endpoint. |                   |
|         |          | 0                                                                                                               | Disable Interrupt |
|         |          | 1                                                                                                               | Enable Interrupt  |

**Table 22-14:** USB\_INTRTXE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------|
| 9       | EP9      | End Point 9 Tx Interrupt Enable.<br>The USB_INTRTXE . EP9 bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                        |
|         |          | 1      Enable Interrupt                                                                                         |
| 8       | EP8      | End Point 8 Tx Interrupt Enable.<br>The USB_INTRTXE . EP8 bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                        |
|         |          | 1      Enable Interrupt                                                                                         |
| 7       | EP7      | End Point 7 Tx Interrupt Enable.<br>The USB_INTRTXE . EP7 bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                        |
|         |          | 1      Enable Interrupt                                                                                         |
| 6       | EP6      | End Point 6 Tx Interrupt Enable.<br>The USB_INTRTXE . EP6 bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                        |
|         |          | 1      Enable Interrupt                                                                                         |
| 5       | EP5      | End Point 5 Tx Interrupt Enable.<br>The USB_INTRTXE . EP5 bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                        |
|         |          | 1      Enable Interrupt                                                                                         |

**Table 22-14:** USB\_INTRTXE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------|
| 4       | EP4      | End Point 4 Tx Interrupt Enable.<br>The <code>USB_INTRTXE.EP4</code> bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                   |
|         |          | 1      Enable Interrupt                                                                                                    |
| 3       | EP3      | End Point 3 Tx Interrupt Enable.<br>The <code>USB_INTRTXE.EP3</code> bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                   |
|         |          | 1      Enable Interrupt                                                                                                    |
| 2       | EP2      | End Point 2 Tx Interrupt Enable.<br>The <code>USB_INTRTXE.EP2</code> bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                   |
|         |          | 1      Enable Interrupt                                                                                                    |
| 1       | EP1      | End Point 1 Tx Interrupt Enable.<br>The <code>USB_INTRTXE.EP1</code> bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                   |
|         |          | 1      Enable Interrupt                                                                                                    |
| 0       | EP0      | End Point 0 Tx Interrupt Enable.<br>The <code>USB_INTRTXE.EP0</code> bit enables the transmit interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                   |
|         |          | 1      Enable Interrupt                                                                                                    |

## Receive Interrupt Enable Register

The `USB_INTRRXE` register enables interrupts for the receive (Rx) endpoints. Enabling an interrupt in this register directs the USB controller to generate an interrupt if the corresponding interrupt pending bit in the `USB_INTRRX` register is set.

USB\_INTRRXE: Receive Interrupt Enable Register - R/W

Reset = 0x0ffe

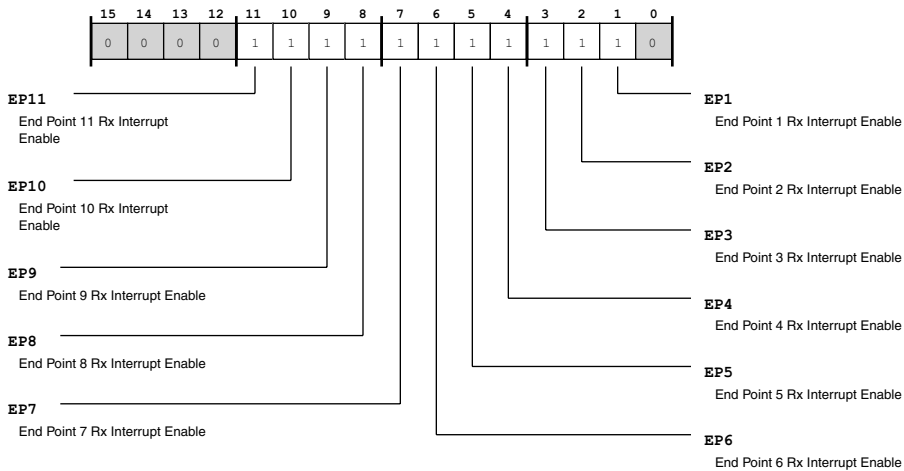


Figure 22-40: USB\_INTRRXE Register Diagram

Table 22-15: USB\_INTRRXE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------|
| 11      | EP11     | End Point 11 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP11</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                    |
|         |          | 1      Enable Interrupt                                                                                                     |
| 10      | EP10     | End Point 10 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP10</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                    |
|         |          | 1      Enable Interrupt                                                                                                     |

**Table 22-15:** USB\_INTRRXE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------|
| 9       | EP9      | End Point 9 Rx Interrupt Enable.<br>The USB_INTRRXE . EP9 bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                       |
|         |          | 1      Enable Interrupt                                                                                        |
| 8       | EP8      | End Point 8 Rx Interrupt Enable.<br>The USB_INTRRXE . EP8 bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                       |
|         |          | 1      Enable Interrupt                                                                                        |
| 7       | EP7      | End Point 7 Rx Interrupt Enable.<br>The USB_INTRRXE . EP7 bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                       |
|         |          | 1      Enable Interrupt                                                                                        |
| 6       | EP6      | End Point 6 Rx Interrupt Enable.<br>The USB_INTRRXE . EP6 bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                       |
|         |          | 1      Enable Interrupt                                                                                        |
| 5       | EP5      | End Point 5 Rx Interrupt Enable.<br>The USB_INTRRXE . EP5 bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                       |
|         |          | 1      Enable Interrupt                                                                                        |

**Table 22-15:** USB\_INTRRXE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------|
| 4       | EP4      | End Point 4 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP4</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                  |
|         |          | 1      Enable Interrupt                                                                                                   |
| 3       | EP3      | End Point 3 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP3</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                  |
|         |          | 1      Enable Interrupt                                                                                                   |
| 2       | EP2      | End Point 2 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP2</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                  |
|         |          | 1      Enable Interrupt                                                                                                   |
| 1       | EP1      | End Point 1 Rx Interrupt Enable.<br>The <code>USB_INTRRXE.EP1</code> bit enables the receive interrupt for this endpoint. |
|         |          | 0      Disable Interrupt                                                                                                  |
|         |          | 1      Enable Interrupt                                                                                                   |

## Common Interrupts Register

The `USB_IRQ` register indicates which interrupts are currently active for USB controller system sources. Note that the USB controller automatically clears this register when it is read.

USB\_IRQ: Common Interrupts Register - R/NW

Reset = 0x00

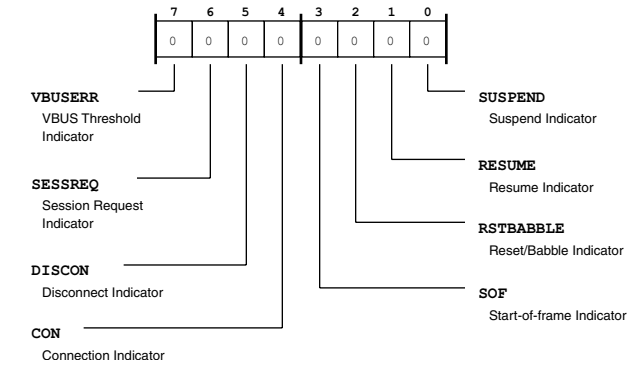


Figure 22-41: USB\_IRQ Register Diagram

Table 22-16: USB\_IRQ Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                  |                   |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 7       | VBUSERR  | VBUS Threshold Indicator.<br>The USB_IRQ.VBUSERR bit indicates whether the USB controller has detected that the VBUS is below the VBUS valid threshold. This bit is valid only when the USB controller is an A device. Note that the USB_IRQ.VBUSERR bit and the USB_VBUS_CTL.DRVINT bit share an interrupt source line. |                   |
|         |          | 0                                                                                                                                                                                                                                                                                                                        | No Interrupt      |
|         |          | 1                                                                                                                                                                                                                                                                                                                        | Interrupt Pending |
| 6       | SESSREQ  | Session Request Indicator.<br>The USB_IRQ.SESSREQ bit indicates whether the USB controller has detected a session request signal. This bit is valid only when the USB controller is an A device.                                                                                                                         |                   |
|         |          | 0                                                                                                                                                                                                                                                                                                                        | No Interrupt      |
|         |          | 1                                                                                                                                                                                                                                                                                                                        | Interrupt Pending |

**Table 22-16:** USB\_IRQ Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                     |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | DISCON    | Disconnect Indicator.<br>The <code>USB_IRQ.DISCON</code> bit indicates whether the USB controller has detected a device disconnect (host mode) or has detected a session end (peripheral mode).                                                                                                                                             |
|         |           | 0 No Interrupt                                                                                                                                                                                                                                                                                                                              |
|         |           | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                         |
| 4       | CON       | Connection Indicator.<br>The <code>USB_IRQ.CON</code> bit indicates whether the USB controller has detected a device connection. This bit is valid only in host mode.                                                                                                                                                                       |
|         |           | 0 No Interrupt                                                                                                                                                                                                                                                                                                                              |
|         |           | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                         |
| 3       | SOF       | Start-of-frame Indicator.<br>The <code>USB_IRQ.SOF</code> bit indicates whether the USB controller has detected a start of frame.                                                                                                                                                                                                           |
|         |           | 0 No Interrupt                                                                                                                                                                                                                                                                                                                              |
|         |           | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                         |
| 2       | RSTBABBLE | Reset/Babble Indicator.<br>The <code>USB_IRQ.RSTBABBLE</code> bit indicates whether the USB controller has detected reset signalling on the bus. In host mode, the USB controller also indicates when the USB controller detects babble. Note that the <code>USB_IRQ.RSTBABBLE</code> bit is only active after the first SOF has been sent. |
|         |           | 0 No Interrupt                                                                                                                                                                                                                                                                                                                              |
|         |           | 1 Interrupt Pending                                                                                                                                                                                                                                                                                                                         |



**Table 22-16:** USB\_IRQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | RESUME   | Resume Indicator.<br>The <code>USB_IRQ.RESUME</code> bit indicates whether the USB controller has detected resume signaling on the bus while the USB controller is in suspend mode.    |
|         |          | 0 No Interrupt                                                                                                                                                                         |
|         |          | 1 Interrupt Pending                                                                                                                                                                    |
| 0       | SUSPEND  | Suspend Indicator.<br>The <code>USB_IRQ.SUSPEND</code> bit indicates whether the USB controller has detected suspend signalling on the bus. This bit is valid only in peripheral mode. |
|         |          | 0 No Interrupt                                                                                                                                                                         |
|         |          | 1 Interrupt Pending                                                                                                                                                                    |

## Common Interrupts Enable Register

The `USB_IEN` register enables interrupts for USB controller system sources. Enabling an interrupt in this register directs the USB controller to generate an interrupt if the corresponding interrupt pending bit in the `USB_IRQ` register is set.

**USB\_IEN: Common Interrupts Enable Register - R/W**

Reset = 0x00

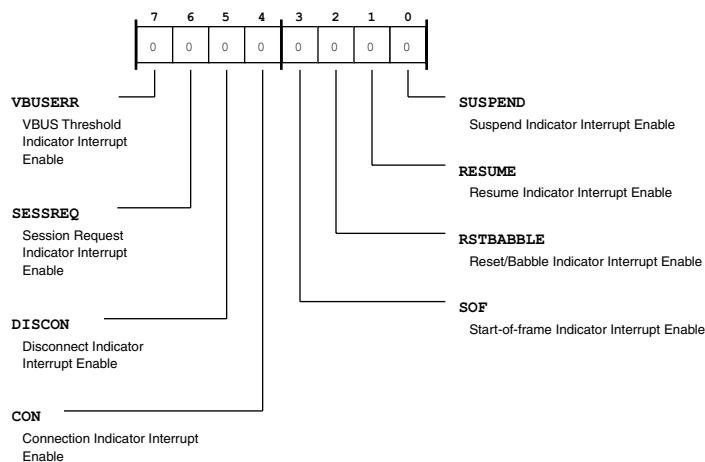


**Figure 22-42:** USB\_IEN Register Diagram

**Table 22-17:** USB\_IEN Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                  |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | VBUSERR   | VBUS Threshold Indicator Interrupt Enable.<br>The <code>USB_IEN.VBUSERR</code> bit enables the <code>USB_IRQ.VBUSERR</code> interrupt.   |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |
| 6       | SESSREQ   | Session Request Indicator Interrupt Enable.<br>The <code>USB_IEN.SESSREQ</code> bit enables the <code>USB_IRQ.SESSREQ</code> interrupt.  |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |
| 5       | DISCON    | Disconnect Indicator Interrupt Enable.<br>The <code>USB_IEN.DISCON</code> bit enables the <code>USB_IRQ.DISCON</code> interrupt.         |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |
| 4       | CON       | Connection Indicator Interrupt Enable.<br>The <code>USB_IEN.CON</code> bit enables the <code>USB_IRQ.CON</code> interrupt.               |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |
| 3       | SOF       | Start-of-frame Indicator Interrupt Enable.<br>The <code>USB_IEN.SOF</code> bit enables the <code>USB_IRQ.SOF</code> interrupt.           |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |
| 2       | RSTBABBLE | Reset/Babble Indicator Interrupt Enable.<br>The <code>USB_IEN.RSTBABBLE</code> bit enables the <code>USB_IRQ.RSTBABBLE</code> interrupt. |
|         |           | 0      Disable Interrupt                                                                                                                 |
|         |           | 1      Enable Interrupt                                                                                                                  |

**Table 22-17:** USB\_IEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| 1       | RESUME   | Resume Indicator Interrupt Enable.<br>The <code>USB_IEN.RESUME</code> bit enables the <code>USB_IRQ.RESUME</code> interrupt.    |
|         |          | 0                      Disable Interrupt                                                                                        |
|         |          | 1                      Enable Interrupt                                                                                         |
| 0       | SUSPEND  | Suspend Indicator Interrupt Enable.<br>The <code>USB_IEN.SUSPEND</code> bit enables the <code>USB_IRQ.SUSPEND</code> interrupt. |
|         |          | 0                      Disable Interrupt                                                                                        |
|         |          | 1                      Enable Interrupt                                                                                         |

## Frame Number Register

The `USB_FRAME` register contains the frame number of the last received frame. The data in this register has bit 10 as the MSB and bit 0 as the LSB.

**USB\_FRAME: Frame Number Register - R/NW**

Reset = 0x0000

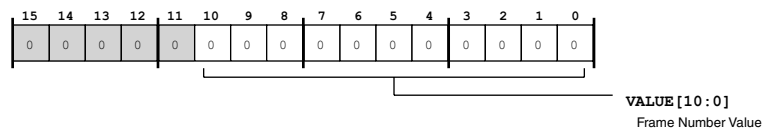


**Figure 22-43:** USB\_FRAME Register Diagram

**Table 22-18:** USB\_FRAME Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10:0    | VALUE    | Frame Number Value.<br>The <code>USB_FRAME.VALUE</code> bits contains the frame number of the last received frame. The data in this field has bit 10 as the MSB and bit 0 as the LSB. |

## Index Register

The `USB_INDEX` register contains an index value for mirrored addressing of USB controller endpoint control and status registers.

There is one set of registers, but they are mirrored at two address locations if the endpoint is selected by the `USB_INDEX` register. An endpoint's register set only appears in the indexed location if the `USB_INDEX` register is written with that endpoint number. You can read/write an endpoint's register in either the directly mapped location which is always visible, or in the indexed location which is only visible if the `USB_INDEX` register is written with the endpoint number. The `USB_INDEX` register and indexed address locations only affect address decoding. For example, loading a 0 into the `USB_INDEX` register selects endpoint 0 access.

The `USB_INDEX` register can be used for indexed access of the directly mapped control/status registers from USB controller address offset 0x100-0x1FF. The endpoint `USB_TXFIFOSZ`, `USB_RXFIFOSZ`, `USB_TXFIFOADDR`, and `USB_RXFIFOADDR` registers always use the `USB_INDEX` register, there is no direct mapping for these endpoint specific registers. The multipoint `USB_MPn_TXFUNCADDR`, `USB_MPn_TXHUBADDR`, `USB_MPn_TXHUBPORT`, `USB_MPn_RXFUNCADDR`, `USB_MPn_RXHUBADDR`, and `USB_MPn_RXHUBPORT` register only have direct mapping, no indexed mapping.

Before accessing an endpoint's control/status registers using the indexed range, write the endpoint number to the `USB_INDEX` register to ensure that the correct control/status registers appear in the indexed range of the memory map.

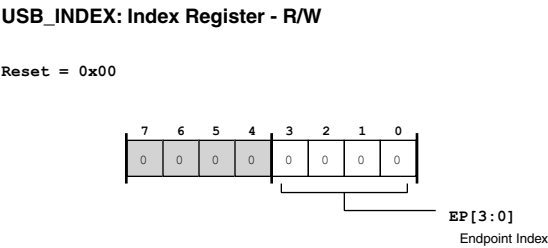


Figure 22-44: `USB_INDEX` Register Diagram

Table 22-19: `USB_INDEX` Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | EP       | Endpoint Index.<br>The <code>USB_INDEX</code> . EP bits selects mirrored access for an endpoint's indexed control and status registers. Valid values for this bit field are 0-11. |

## Testmode Register

The `USB_TESTMODE` register places the USB controller into test mode state and can also put the USB controller into one of the test modes for high-speed operation. For more information about these modes, see the USB 2.0 specification.

Note that the `USB_TESTMODE` register is not used in normal operation. Only one of the test mode (bits 0-6) selection bits may be set at a time.

**USB\_TESTMODE: Testmode Register - R/W**

Reset = 0x00

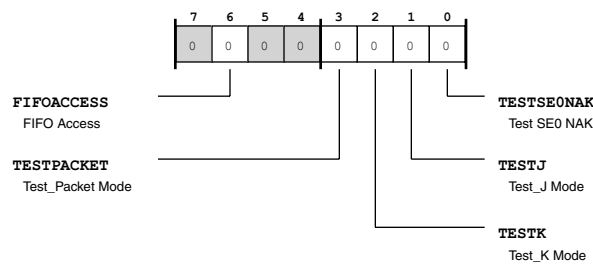


**Figure 22-45:** USB\_TESTMODE Register Diagram

**Table 22-20:** USB\_TESTMODE Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | FIFOACCESS | FIFO Access.<br>The <code>USB_TESTMODE</code> bit directs the USB controller to transfer the packet in the endpoint 0 Tx FIFO to the endpoint 0 Rx FIFO. The bit is cleared automatically.                                                                                                                                                                                                                                                               |
| 3       | TESTPACKET | Test_Packet Mode.<br>The <code>USB_TESTMODE.TESTPACKET</code> bit selects Test_Packet test mode, which applies only when the USB controller in high speed mode. In this mode, the USB controller repetitively transmits on the bus a 53-byte test packet, whose form is defined in the USB 2.0 Specification, Section 7.1.20. Note that the test packet has a fixed format and must be loaded into the endpoint 0 FIFO before this test mode is entered. |
| 2       | TESTK      | Test_K Mode.<br>The <code>USB_TESTMODE.TESTK</code> bit selects Test_K test mode. In this mode, the USB controller transmits a continuous K on the bus.                                                                                                                                                                                                                                                                                                  |

Table 22-20: USB\_TESTMODE Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                     |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | TESTJ      | Test_J Mode.<br>The USB_TESTMODE.TESTJ bit selects Test_J test mode. In this mode, the USB controller transmits a continuous J on the bus.                                                                                                                  |
| 0       | TESTSE0NAK | Test SE0 NAK.<br>The USB_TESTMODE.TESTSE0NAK bit selects Test_SE0_NAK test mode, which applies only when the USB controller in high speed mode. In this mode, the USB controller remains in high-speed mode, but responds to any valid IN token with a NAK. |

FIFO Byte (8-Bit) Register

Writes to the USB\_FIFOBn register go to the endpoint Tx FIFO and reads from the USB\_FIFOBn register come from the endpoint Rx FIFO. The USB\_FIFOBn, USB\_FIFOHn, and USB\_FIFOn registers are one and the same. These registers exist at the same address. Typically, programs should load and unload the FIFO using word (USB\_FIFOn register) writes and reads, which are more efficient. Only if the USB packet is a non-word (4-byte) size should the program use a half-word (USB\_FIFOHn register) or byte (USB\_FIFOBn register) read or write at the end when loading or unloading the FIFO.

Note that (for correct USB controller operation) programs should not mix byte, half-word, or word accesses, except for the last few bytes if the size of the packet is odd (not a multiple of the size they were using).

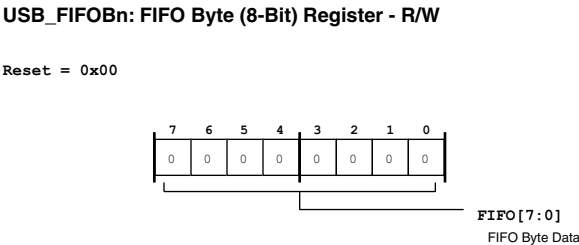


Figure 22-46: USB\_FIFOBn Register Diagram

Table 22-21: USB\_FIFOBn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------|
| 7:0     | FIFO     | FIFO Byte Data.<br>The USB_FIFOBn.FIFO bits provide byte access to the USB Tx and Rx endpoint FIFOs. |

## FIFO Half-Word (16-Bit) Register

Writes to the USB\_FIFOHn register go to the endpoint Tx FIFO and reads from the USB\_FIFOHn register come from the endpoint Rx FIFO. The USB\_FIFOBn, USB\_FIFOHn, and USB\_FIFOn registers are one and the same. These registers exist at the same address. Typically, programs should load and unload the FIFO using word (USB\_FIFOn register) writes and reads, which are more efficient. Only if the USB packet is a non-word (4-byte) size should the program use a half-word (USB\_FIFOHn register) or byte (USB\_FIFOBn register) read or write at the end when loading or unloading the FIFO.

Note that (for correct USB controller operation) programs should not mix byte, half-word, or word accesses, except for the last few bytes if the size of the packet is odd (not a multiple of the size they were using).

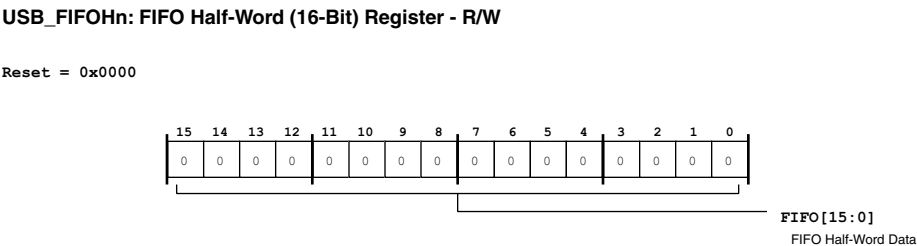


Figure 22-47: USB\_FIFOHn Register Diagram

Table 22-22: USB\_FIFOHn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------|
| 15:0    | FIFO     | FIFO Half-Word Data.<br>The USB_FIFOHn.FIFO bits provide half-word access to the USB Tx and Rx endpoint FIFOs. |

## FIFO Word (32-Bit) Register

Writes to the USB\_FIFOn register go to the endpoint Tx FIFO and reads from the USB\_FIFOn register come from the endpoint Rx FIFO. The USB\_FIFOBn, USB\_FIFOHn, and USB\_FIFOn registers are one and the same. These registers exist at the same address. Typically, programs should load and unload the FIFO using word (USB\_FIFOn register) writes and reads, which are more efficient. Only if the USB packet is a non-word (4-byte) size should the program use a half-word (USB\_FIFOHn register) or byte (USB\_FIFOBn register) read or write at the end when loading or unloading the FIFO.

Note that (for correct USB controller operation) programs should not mix byte, half-word, or word accesses, except for the last few bytes if the size of the packet is odd (not a multiple of the size they were using).

USB\_FIFOn: FIFO Word (32-Bit) Register - R/W

Reset = 0x0000 0000

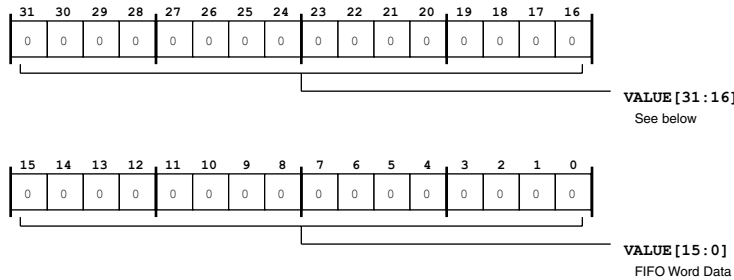


Figure 22-48: USB\_FIFOn Register Diagram

Table 22-23: USB\_FIFOn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | FIFO Word Data.<br>The USB_FIFOn.VALUE bits provide word access to the USB Tx and Rx endpoint FIFOs. |

Device Control Register

The USB\_DEV\_CTL register selects whether the USB controller is operating in peripheral mode or in host mode and is used for controlling and monitoring the VBUS line.

USB\_DEV\_CTL: Device Control Register - R/W

Reset = 0x00

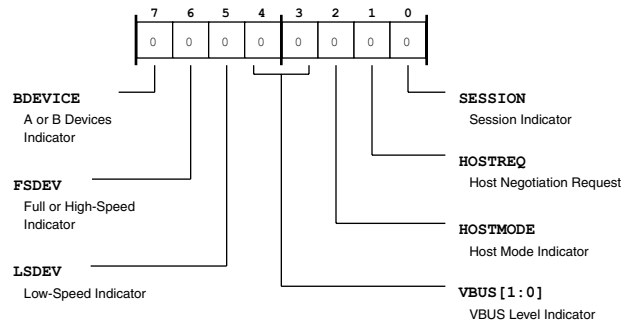


Figure 22-49: USB\_DEV\_CTL Register Diagram



**Table 22-24:** USB\_DEV\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | BDEVICE  | A or B Devices Indicator.<br>The <code>USB_DEV_CTL.BDEVICE</code> bit indicates whether the USB controller is operating as the A device or the B device. This bit is only valid while a session is in progress.                                                                                                                            |
|         |          | 0      A Device Detected                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      B Device Detected                                                                                                                                                                                                                                                                                                                   |
| 6       | FSDEV    | Full or High-Speed Indicator.<br>The <code>USB_DEV_CTL.FSDEV</code> bit is set when a full-speed or high-speed device is detected being connected to the port. High speed devices are distinguished from full-speed by checking for high-speed chirps when the device detects a USB controller reset. This bit is only valid in host mode. |
|         |          | 0      Not Detected                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Full or High Speed Detected                                                                                                                                                                                                                                                                                                         |
| 5       | LSDEV    | Low-Speed Indicator.<br>The <code>USB_DEV_CTL.LSDEV</code> bit is set when a low-speed device is detected being connected to the port. This bit is only valid in host mode.                                                                                                                                                                |
|         |          | 0      Not Detected                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Low Speed Detected                                                                                                                                                                                                                                                                                                                  |
| 4:3     | VBUS     | VBUS Level Indicator.<br>The <code>USB_DEV_CTL.VBUS</code> bits indicated the current VBUS level.                                                                                                                                                                                                                                          |
|         |          | 0      Below SessionEnd                                                                                                                                                                                                                                                                                                                    |
|         |          | 1      Above SessionEnd, below AValid                                                                                                                                                                                                                                                                                                      |
|         |          | 2      Above AValid, below VBUSValid                                                                                                                                                                                                                                                                                                       |
|         |          | 3      Above VBUSValid                                                                                                                                                                                                                                                                                                                     |

**Table 22-24:** USB\_DEV\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | HOSTMODE | Host Mode Indicator.<br>The <code>USB_DEV_CTL.HOSTMODE</code> bit is set when the USB controller is acting as a host.                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0 Peripheral Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Host Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 1       | HOSTREQ  | Host Negotiation Request.<br>When the <code>USB_DEV_CTL.HOSTREQ</code> bit is set, the USB controller initiates the host negotiation when Suspend mode is entered. This bit is cleared when host negotiation is completed. The <code>USB_DEV_CTL.HOSTREQ</code> bit applies when the USB controller is operating as a B device only.                                                                                                                                                                                       |
|         |          | 0 No Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 Place Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 0       | SESSION  | Session Indicator.<br>When operating as an A device, the <code>USB_DEV_CTL.SESSION</code> is set or cleared by the processor core to start or end a session. When operating as a B device, the <code>USB_DEV_CTL.SESSION</code> bit is set or cleared by the USB controller when a session starts or ends. This bit is also set by the processor core to initiate the session request protocol. When the USB controller is in Suspend mode, the bit may be cleared by the processor core to perform a software disconnect. |
|         |          | 0 Not Detected                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Detected Session                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

## Transmit FIFO Size Register

The `USB_TXFIFOSZ` register defines the maximum amount of data that can be transferred through the selected transmit endpoint in a single frame. When setting this value, you must consider the constraints placed by the USB specification on packet sizes for bulk, interrupt and isochronous transactions in full-speed operations. This register provides indexed access to the FIFO (packet) size selection for each Tx endpoint (except endpoint 0).

USB\_TXFIFOSZ: Transmit FIFO Size Register - R/W

Reset = 0x00

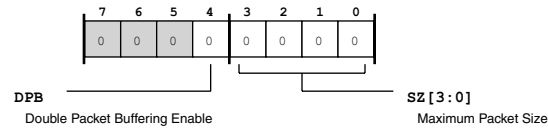


Figure 22-50: USB\_TXFIFOSZ Register Diagram

Table 22-25: USB\_TXFIFOSZ Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                               |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DPB      | Double Packet Buffering Enable.<br>The USB_TXFIFOSZ.DPB bit enables double packet buffering, doubling the FIFO (packet) size selected with the USB_TXFIFOSZ.SZ field. |
|         |          | 0 Single Packet Buffering                                                                                                                                             |
|         |          | 1 Double Packet Buffering                                                                                                                                             |

**Table 22-25:** USB\_TXFIFOSZ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | SZ       | <p>Maximum Packet Size.</p> <p>The <code>USB_TXFIFOSZ.SZ</code> bits select the maximum FIFO (packet) size according to the formula:</p> $\text{FIFOSZ} = 2^{(\text{SZ}+3)}$ <p>. If the <code>USB_TXFIFOSZ.DPB</code> is cleared, the FIFO size is FIFOSZ from this formula. If the <code>USB_TXFIFOSZ.DPB</code> is set, the FIFO is twice this size.</p> <p>For each enumeration value, the enumerations descriptions show the packet size (<code>PktSz=</code>), the FIFO size if <code>DPB=0</code> (<code>DPB0=</code>), and the FIFO size if <code>DPB=1</code> (<code>DPB1=</code>); these values are in bytes.</p> |
|         |          | 0      PktSz=8, DPB0=8, DPB1=16                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1      PktSz=16, DPB0=16, DPB1=32                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2      PktSz=32, DPB0=32, DPB1=64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 3      PktSz=64, DPB0=64, DPB1=128                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 4      PktSz=128, DPB0=128, DPB1=256                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 5      PktSz=256, DPB0=256, DPB1=512                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 6      PktSz=512, DPB0=512, DPB1=1024                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 7      PktSz=1024, DPB0=1024, DPB1=2048                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 8      PktSz=2048, DPB0=2048, DPB1=4096                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 9      PktSz=4096, DPB0=4096, DPB1=8192                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

## Receive FIFO Size Register

The `USB_RXFIFOSZ` register defines the maximum amount of data that can be transferred through the selected receive endpoint in a single frame. When setting this value, you must consider the constraints placed by the USB specification on packet sizes for bulk, interrupt and isochronous transactions in full-speed operations. This register provides indexed access to the FIFO (packet) size selection for each Rx endpoint (except endpoint 0).

Note that a value greater than the maximum allowed of 1023 for full-speed USB controller operation produces unpredictable results.

Also note that the value written to this register should match the programmed maximum individual packet size (MaxPktSize) of the standard endpoint descriptor for the associated endpoint (see Universal Serial Bus Specification Revision 2.0, Chapter 9). A mismatch could cause unexpected results. The total amount of data represented by the value written to this register must not exceed the Rx FIFO size, and should not exceed half the FIFO size if double-buffering is required.

USB\_RXFIFOSZ: Receive FIFO Size Register - R/W

Reset = 0x00

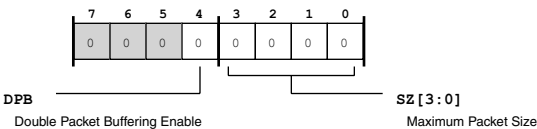


Figure 22-51: USB\_RXFIFOSZ Register Diagram

Table 22-26: USB\_RXFIFOSZ Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                               |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DPB      | Double Packet Buffering Enable.<br>The USB_RXFIFOSZ.DPB bit enables double packet buffering, doubling the FIFO (packet) size selected with the USB_RXFIFOSZ.SZ field. |
|         |          | 0 Single Packet Buffering                                                                                                                                             |
|         |          | 1 Double Packet Buffering                                                                                                                                             |

**Table 22-26:** USB\_RXFIFOSZ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | SZ       | <p>Maximum Packet Size.</p> <p>The <code>USB_RXFIFOSZ.SZ</code> bits select the maximum FIFO (packet) size according to the formula:</p> $\text{FIFOSZ} = 2^{(\text{SZ}+3)}$ <p>. If the <code>USB_RXFIFOSZ.DPB</code> is cleared, the FIFO size is FIFOSZ from this formula. If the <code>USB_RXFIFOSZ.DPB</code> is set, the FIFO is twice this size.</p> <p>For each enumeration value, the enumerations descriptions show the packet size (<code>PktSz=</code>), the FIFO size if <code>DPB=0</code> (<code>DPB0=</code>), and the FIFO size if <code>DPB=1</code> (<code>DPB1=</code>); these values are in bytes.</p> |
|         |          | 0 PktSz=8, DPB0=8, DPB1=16                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 PktSz=16, DPB0=16, DPB1=32                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 2 PktSz=32, DPB0=32, DPB1=64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 PktSz=64, DPB0=64, DPB1=128                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 4 PktSz=128, DPB0=128, DPB1=256                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 5 PktSz=256, DPB0=256, DPB1=512                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 6 PktSz=512, DPB0=512, DPB1=1024                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 7 PktSz=1024, DPB0=1024, DPB1=2048                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 8 PktSz=2048, DPB0=2048, DPB1=4096                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 9 PktSz=4096, DPB0=4096, DPB1=8192                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

## Transmit FIFO Address Register

The `USB_TXFIFOADDR` sets the start address for the selected Tx FIFO for endpoints 1-11. There is one transmit FIFO address register for each endpoint, except endpoint 0. The `USB_TXFIFOADDR` register is indexed and selected by the `USB_INDEX` register. Note that the endpoint 0 FIFO has a fixed 64-byte size and is always located at address 0.

USB\_TXFIFOADDR: Transmit FIFO Address Register - R/W

Reset = 0x0000

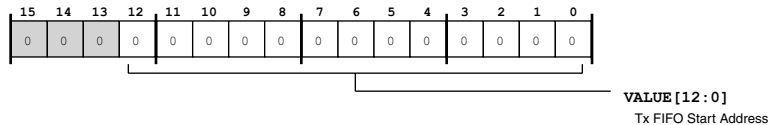


Figure 22-52: USB\_TXFIFOADDR Register Diagram

Table 22-27: USB\_TXFIFOADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12:0    | VALUE    | <p>Tx FIFO Start Address.</p> <p>The USB_TXFIFOADDR.VALUE bits hold the start address of the selected endpoint FIFO (selected with the USB_INDEX register) in units of 8 bytes, according to the formula:</p> <p>FIFO address = USB_TXFIFOADDR.VALUE * 8</p> |

Receive FIFO Address Register

The USB\_RXFIFOADDR sets the start address for the selected Rx FIFO for endpoints 1-11. There is one receive FIFO address register for each endpoint, except endpoint 0. The USB\_RXFIFOADDR register is indexed and selected by the USB\_INDEX register. Note that the endpoint 0 FIFO has a fixed 64-byte size and is always located at address 0.

USB\_RXFIFOADDR: Receive FIFO Address Register - R/W

Reset = 0x0000

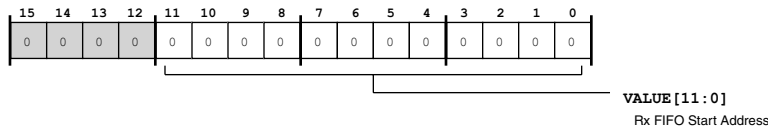


Figure 22-53: USB\_RXFIFOADDR Register Diagram

Table 22-28: USB\_RXFIFOADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:0    | VALUE    | Rx FIFO Start Address.<br>The <code>USB_RXFIFOADDR.VALUE</code> bits hold the start address of the selected endpoint FIFO (selected with the <code>USB_INDEX</code> register) in units of 8 bytes, according to the formula:<br>$\text{FIFO address} = \text{USB\_RXFIFOADDR.VALUE} * 8$ |

Endpoint Information Register

The `USB_EPINFO` register allows read-back of the number of Tx and Rx endpoints available

USB\_EPINFO: Endpoint Information Register - R/NW

Reset = 0xcc

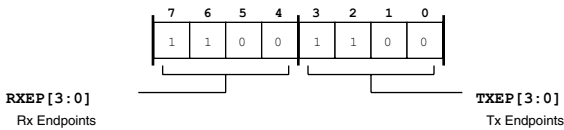


Figure 22-54: USB\_EPINFO Register Diagram

Table 22-29: USB\_EPINFO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------|
| 7:4     | RXEP     | Rx Endpoints.<br>The <code>USB_EPINFO.RXEP</code> bits indicate the number of receive endpoints, excluding EP0.  |
| 3:0     | TXEP     | Tx Endpoints.<br>The <code>USB_EPINFO.TXEP</code> bits indicate the number of transmit endpoints, excluding EP0. |



## RAM Information Register

The `USB_RAMINFO` register provides information about the width of the USB controller RAM.

**USB\_RAMINFO: RAM Information Register - R/NW**

Reset = 0x8c

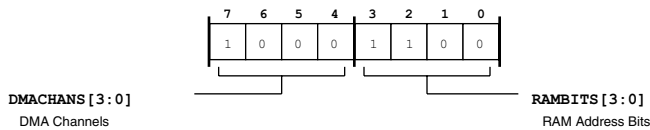


**Figure 22-55:** USB\_RAMINFO Register Diagram

**Table 22-30:** USB\_RAMINFO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | DMACHANS | DMA Channels.<br>The <code>USB_RAMINFO.DMACHANS</code> bits indicate the number of DMA channels.                                                                                                                                                                 |
| 3:0     | RAMBITS  | RAM Address Bits.<br>The <code>USB_RAMINFO.RAMBITS</code> bits indicate the number of RAM address bits. The USB controller FIFO RAM is 32-bits wide. The number of bytes in the FIFO RAM may be calculated from the formula:<br>$RAM\_bytes = 2^{(RAM\_Bits+2)}$ |

## Link Information Register

The `USB_LINKINFO` register specifies the PHY-related delays.

**USB\_LINKINFO: Link Information Register - R/W**

Reset = 0x5c

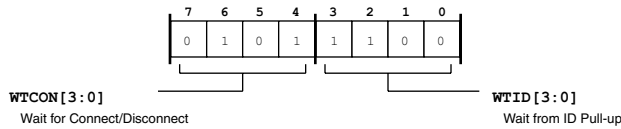


**Figure 22-56:** USB\_LINKINFO Register Diagram

Table 22-31: USB\_LINKINFO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | WTCON    | Wait for Connect/Disconnect.<br>The USB_LINKINFO.WTCON bits set the wait to be applied to allow for the users connect or disconnect filter in units of 533.3ns. The default settings corresponds to 2.667us                                                                                                                                                                                                                                                                        |
| 3:0     | WTID     | Wait from ID Pull-up.<br>The USB_LINKINFO.WTID bits set the delay to be applied from IDPULLUP being asserted to IDDIG being considered valid in units of 4.3690ms. The default corresponds to 52.43ms UTMI+ spec says 50ms min. OTG spec does not have timing requirements (it doesn't assume a programmable pull-up that is only sampled during session start). Micro-USB cable spec says that the ID pin is greater than 10 Ohms when shorted and less than 100k Ohms when open. |

VBUS Pulse Length Register

The USB\_VPLEN register defines the duration of the VBUS pulsing charge for SRP initiation.

USB\_VPLEN: VBUS Pulse Length Register - R/W

Reset = 0x3c

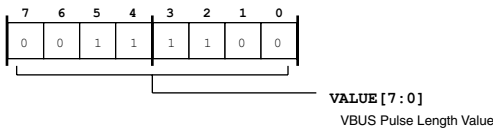


Figure 22-57: USB\_VPLEN Register Diagram

Table 22-32: USB\_VPLEN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | VBUS Pulse Length Value.<br>The USB_VPLEN.VALUE bits sets the duration of the VBUS pulsing charge in units of 546.1us. The default setting corresponds to 32.77ms. Note that VBUS pulsing was removed in the OTG specification v2.0, section 5.1.4. |

## High-Speed EOF 1 Register

The `USB_HS_EOF1` register defines the minimum time gap allowed between the start of the last transaction and the end of frame for high-speed transactions.

USB\_HS\_EOF1: High-Speed EOF 1 Register - R/W

Reset = 0x80

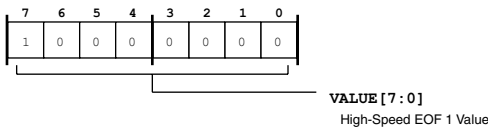


Figure 22-58: USB\_HS\_EOF1 Register Diagram

Table 22-33: USB\_HS\_EOF1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | High-Speed EOF 1 Value.<br>The <code>USB_HS_EOF1.VALUE</code> sets the time before end of frame to stop beginning new transactions (in units of 133.3ns) for high-speed transactions. The default setting corresponds to 17.07us. |

## Full-Speed EOF 1 Register

The `USB_FS_EOF1` register defines the minimum time gap allowed between the start of the last transaction and the end of frame for full-speed transactions.

USB\_FS\_EOF1: Full-Speed EOF 1 Register - R/W

Reset = 0x77

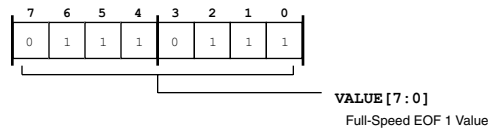


Figure 22-59: USB\_FS\_EOF1 Register Diagram

Table 22-34: USB\_FS\_EOF1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | Full-Speed EOF 1 Value.<br>The <code>USB_FS_EOF1.VALUE</code> bits set the time before end of frame to stop beginning new transactions (in units of 533.3ns) for full-speed transactions. The default setting corresponds to 63.46us. |

Low-Speed EOF 1 Register

The `USB_LS_EOF1` register defines the minimum time gap allowed between the start of the last transaction and the end of frame for low-speed transactions.

USB\_LS\_EOF1: Low-Speed EOF 1 Register - R/W  
  
Reset = 0x72

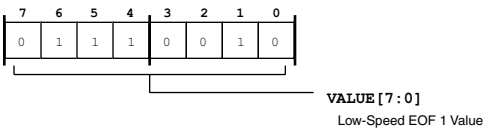


Figure 22-60: USB\_LS\_EOF1 Register Diagram

Table 22-35: USB\_LS\_EOF1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | Low-Speed EOF 1 Value.<br>The <code>USB_LS_EOF1.VALUE</code> bits set the time before end of frame to stop beginning new transactions (in units of 1.067us) for low-speed transactions. The default setting corresponds to 121.6us. |

Software Reset Register

The `USB_SOFT_RST` register provides reset controls for the USB controller CLK domain and XCLK domain. The USB controller PHY operates in the controller's XCLK domain, and the USB controller interface to the processor core operates in the controller's CLK domain. Note that for correct operation, both of the reset control bits (`USB_SOFT_RST.RST` and `USB_SOFT_RST.RSTX`) should always be asserted simultaneously.

USB\_SOFT\_RST: Software Reset Register - R/W

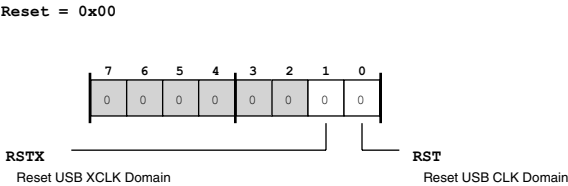


Figure 22-61: USB\_SOFT\_RST Register Diagram

Table 22-36: USB\_SOFT\_RST Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                        |                       |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 1       | RSTX     | Reset USB XCLK Domain.<br>The USB_SOFT_RST.RSTX bit resets logic in the USB XCLK domain. This bit is self clearing. Note that this bit should always be asserted simultaneously with the USB_SOFT_RST.RST bit. |                       |
|         |          | 0                                                                                                                                                                                                              | No Reset              |
|         |          | 1                                                                                                                                                                                                              | Reset USB XCLK Domain |
| 0       | RST      | Reset USB CLK Domain.<br>The USB_SOFT_RST.RST bit resets logic in the USB CLK domain. This bit is self clearing. Note that this bit should always be asserted simultaneously with the USB_SOFT_RST.RSTX bit.   |                       |
|         |          | 0                                                                                                                                                                                                              | No Reset              |
|         |          | 1                                                                                                                                                                                                              | Reset USB CLK Domain  |

MPn Transmit Function Address Register

The USB\_MPn\_TXFUNCADDR register specifies the transmit endpoint's target address in host mode. This register is not used in device mode. Note that the USB\_MPn\_TXFUNCADDR register must be setup for EP0. (The USB\_MPn\_RXFUNCADDR register does not exist for EP0.)

USB\_MPN\_TXFUNCADDR: MPn Transmit Function Address Register - R/W

Reset = 0x00

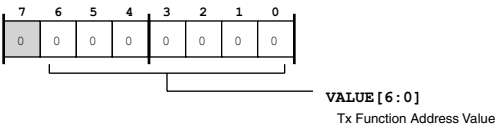


Figure 22-62: USB\_MPN\_TXFUNCADDR Register Diagram

Table 22-37: USB\_MPN\_TXFUNCADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Tx Function Address Value.<br>The USB_MPN_TXFUNCADDR.VALUE bits hold the address of the target device for this endpoint. |

MPn Transmit Hub Address Register

The USB\_MPN\_TXHUBADDR register specifies the hub address of the endpoint in host mode. This register is not used in device mode. Note that this register only needs to be programmed when a full-speed or low-speed device is connected to a high-speed hub, which carries out the necessary transaction translation. Also note that EP0 only uses the USB\_MPN\_TXHUBADDR register. (The USB\_MPN\_RXHUBADDR register does not exist for EP0.)

USB\_MPN\_TXHUBADDR: MPn Transmit Hub Address Register - R/W

Reset = 0x00

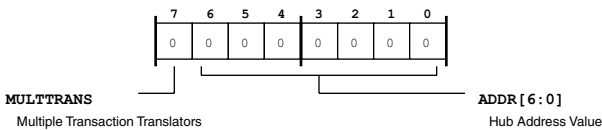


Figure 22-63: USB\_MPN\_TXHUBADDR Register Diagram

**Table 22-38:** USB\_MPn\_TXHUBADDR Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                 |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 7       | MULTTRANS | Multiple Transaction Translators.<br>The USB_MPn_TXHUBADDR.MULTTRANS bit should be set if the hub has multiple transaction translators. |
|         |           | 0                      Single Transaction Translator                                                                                    |
|         |           | 1                      Multiple Transaction Translators                                                                                 |
| 6:0     | ADDR      | Hub Address Value.<br>The USB_MPn_TXHUBADDR.ADDR bits hold the address of the hub to which this device is connected.                    |

MPn Transmit Hub Port Register

The USB\_MPn\_TXHUBPORT register specifies the hub port for full-speed and low-speed endpoints in host mode. This register is not used in device mode. The USB\_MPn\_TXHUBPORT register lets the USB controller support SPLIT transactions. EP0 only uses the USB\_MPn\_TXHUBPORT register. (The USB\_MPn\_RXHUBPORT register does not exist for EP0.)

USB\_MPn\_TXHUBPORT: MPn Transmit Hub Port Register - R/W

Reset = 0x00

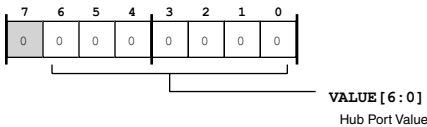


**Figure 22-64:** USB\_MPn\_TXHUBPORT Register Diagram

**Table 22-39:** USB\_MPn\_TXHUBPORT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Hub Port Value.<br>The USB_MPn_TXHUBPORT.VALUE bits hold the hub port value of the target device for this endpoint. |

## MPn Receive Function Address Register

The `USB_MPn_RXFUNCADDR` register specifies the receive endpoint's target address in host mode. This register is not used in device mode. Note that the `USB_MPn_RXFUNCADDR` register does not exist for EP0.

**USB\_MPn\_RXFUNCADDR: MPn Receive Function Address Register - R/W**  
  
**Reset = 0x00**

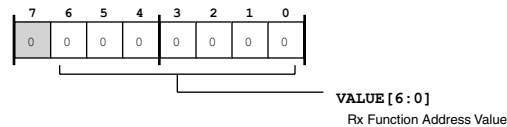


**Figure 22-65:** USB\_MPn\_RXFUNCADDR Register Diagram

**Table 22-40:** USB\_MPn\_RXFUNCADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Rx Function Address Value.<br>The <code>USB_MPn_RXFUNCADDR.VALUE</code> bits hold the address of the target device for this endpoint. |

## MPn Receive Hub Address Register

The `USB_MPn_RXHUBADDR` register specifies the hub address of the endpoint in host mode. This register is not used in device mode. Note that this register only needs to be programmed when a full-speed or low-speed device is connected to a high-speed hub, which carries out the necessary transaction translation. Note that the `USB_MPn_RXHUBADDR` register does not exist for EP0.

**USB\_MPn\_RXHUBADDR: MPn Receive Hub Address Register - R/W**  
  
**Reset = 0x00**

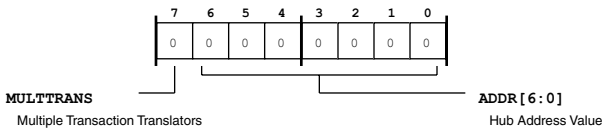


**Figure 22-66:** USB\_MPn\_RXHUBADDR Register Diagram



**Table 22-41:** USB\_MPn\_RXHUBADDR Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                 |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 7       | MULTTRANS | Multiple Transaction Translators.<br>The USB_MPn_RXHUBADDR.MULTTRANS bit should be set if the hub has multiple transaction translators. |
|         |           | 0 Single Transaction Translator                                                                                                         |
|         |           | 1 Multiple Transaction Translators                                                                                                      |
| 6:0     | ADDR      | Hub Address Value.<br>The USB_MPn_RXHUBADDR.ADDR bits hold the address of the hub to which this device is connected.                    |

## MPn Receive Hub Port Register

The USB\_MPn\_RXHUBPORT register specifies the hub port for full-speed and low-speed endpoints in host mode. This register is not used in device mode. The USB\_MPn\_RXHUBPORT register lets the USB controller support SPLIT transactions. Note that the USB\_MPn\_RXHUBPORT register does not exist for EP0.

**USB\_MPn\_RXHUBPORT: MPn Receive Hub Port Register - R/W**

Reset = 0x00

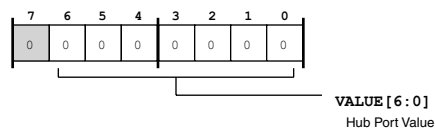


**Figure 22-67:** USB\_MPn\_RXHUBPORT Register Diagram

**Table 22-42:** USB\_MPn\_RXHUBPORT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Hub Port Value.<br>The USB_MPn_RXHUBPORT.VALUE bits hold the hub port value of the target device for this endpoint. |

## EPn Transmit Maximum Packet Length Register

The USB\_EPn\_TXMAXP register defines the maximum amount of data that can be transferred through the selected transmit endpoint in a single frame. When setting this value, you must consider the constraints

placed by the USB specification on packet sizes for bulk, interrupt and isochronous transactions in full-speed operations. The `USB_EPn_TXMAXP` register provides indexed access to the maximum packet length register for each Tx endpoint, except endpoint 0.

**USB\_EPn\_TXMAXP: EPn Transmit Maximum Packet Length Register - R/W**

Reset = 0x0000

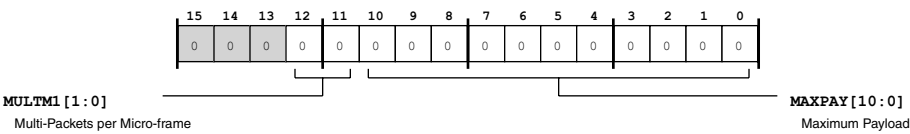


**Figure 22-68:** USB\_EPn\_TXMAXP Register Diagram

**Table 22-43:** USB\_EPn\_TXMAXP Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12:11   | MULTM1   | Multi-Packets per Micro-frame.<br>The <code>USB_EPn_TXMAXP.MULTM1</code> bits select the number of high-speed high-bandwidth isochronous or interrupt packets that may be transferred in a microframe. The valid number of packets per microframe is 1-3 which corresponds to settings 0-2. If this field is not zero, the USB controller splits the FIFO data into multiple packets less than or equal to the maximum payload size.                                                                                                                                                                                                             |
| 10:0    | MAXPAY   | Maximum Payload.<br>The <code>USB_EPn_TXMAXP.MAXPAY</code> bits select the maximum number of bytes that may be transferred per transaction. This field can be up to 1024 but is subject to constraints by the USB specification based on endpoint mode and speed. This field should not exceed the FIFO size for the endpoint, or half the FIFO size if double buffering is used. This value should match the <code>wMaxPacketSize</code> field of the standard endpoint descriptor (USB 2.0 spec, section 9). The <code>USB_EPn_TXMAXP.MAXPAY</code> bits must be set to an even number of bytes for proper interrupt generation in DMA mode 1. |

EP0 Configuration and Status (Host) Register

The `USB_EP0_CSRn_H` register provides control and status bits for endpoint 0 in host mode. Note that some bits may be set to clear automatically.

**USB\_EP0\_CSRn\_H: EP0 Configuration and Status (Host) Register - R/W**

Reset = 0x0000

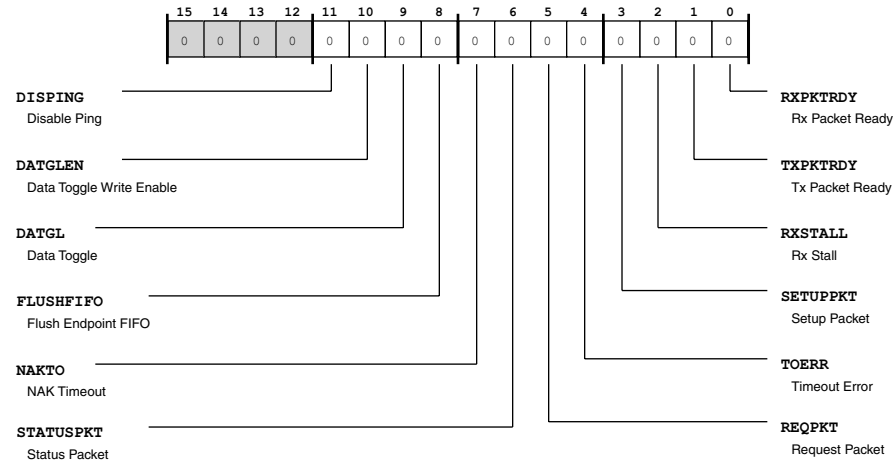


**Figure 22-69:** USB\_EP0\_CSRn\_H Register Diagram

**Table 22-44:** USB\_EP0\_CSRn\_H Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | DISPING  | Disable Ping.<br>The <code>USB_EP0_CSRn_H.DISPING</code> bit disables (in host mode) high-speed PING tokens for the data and status phases of a control transfer.                                                                                                       |
|         |          | 0 Issue PING tokens                                                                                                                                                                                                                                                     |
|         |          | 1 Do not issue PING                                                                                                                                                                                                                                                     |
| 10      | DATGLEN  | Data Toggle Write Enable.<br>The <code>USB_EP0_CSRn_H.DATGLEN</code> bit enables (in host mode) the USB controller to write the current state of the endpoint 0 <code>USB_EP0_CSRn_H.DATGL</code> bit. This bit is automatically cleared once the new value is written. |
|         |          | 0 Disable Write to DATGL                                                                                                                                                                                                                                                |
|         |          | 1 Enable Write to DATGL                                                                                                                                                                                                                                                 |

**Table 22-44:** USB\_EP0\_CSRn\_H Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | DATGL     | Data Toggle.<br>The USB_EP0_CSRn_H.DATGL bit indicates (in host mode) the current state of the endpoint 0 data toggle. If D10 is high, this bit may be written with the required setting of the data toggle. If D10 is low, any value written to this bit is ignored. This bit is only used in host mode.                                                                                                           |
|         |           | 0 DATA0 is Set                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 1 DATA1 is Set                                                                                                                                                                                                                                                                                                                                                                                                      |
| 8       | FLUSHFIFO | Flush Endpoint FIFO.<br>The USB_EP0_CSRn_H.FLUSHFIFO bit directs (in host mode) the USB controller to flush data from the endpoint 0 FIFO and clear the USB_EP0_CSRn_H.TXPKTRDY and USB_EP0_CSRn_H.RXPKTRDY bits. The USB_EP0_CSRn_H.FLUSHFIFO bit should only be set if the USB_EP0_CSRn_H.TXPKTRDY and USB_EP0_CSRn_H.RXPKTRDY bits are set. Note that setting this bit at other times may cause data corruption. |
|         |           | 0 No Flush                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Flush Endpoint FIFO                                                                                                                                                                                                                                                                                                                                                                                               |
| 7       | NAKTO     | NAK Timeout.<br>The USB_EP0_CSRn_H.NAKTO bit indicates (in host mode) when endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USB_EP0_NAKLIMITn register. The processor core should clear this bit to allow the endpoint to continue.                                                                                                                                  |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |           | 1 Endpoint Halted (NAK Timeout)                                                                                                                                                                                                                                                                                                                                                                                     |

**Table 22-44:** USB\_EP0\_CSRn\_H Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                     |                            |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| 6       | STATUSPKT | Status Packet.<br>The USB_EP0_CSRn_H.STATUSPKT bit directs (in host mode) the USB controller to perform a status stage transaction. This bit is set at the same time as the USB_EP0_CSRn_H.TXPKTRDY and USB_EP0_CSRn_H.RXPKTRDY. Setting this bit ensures that the data toggle is set to 1 so that a DATA1 packet is used for the status stage transaction. |                            |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                           | No Request                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                           | Request Status Transaction |
| 5       | REQPKT    | Request Packet.<br>The USB_EP0_CSRn_H.REQPKT bit directs (in host mode) the USB controller to request an IN transaction. This bit is cleared when USB_EP0_CSRn_H.RXPKTRDY is set.                                                                                                                                                                           |                            |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                           | No Request                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                           | Send IN Tokens to Device   |
| 4       | TOERR     | Timeout Error.<br>The USB_EP0_CSRn_H.TOERR bit indicates (in host mode) when three attempts have been made to perform a transaction with no response from the peripheral. The processor core should clear this bit. An interrupt is generated when this bit is set.                                                                                         |                            |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                           | No Status                  |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                           | Timeout Error              |
| 3       | SETUPPKT  | Setup Packet.<br>The USB_EP0_CSRn_H.SETUPPKT bit directs (in host mode) the USB controller to send a SETUP token instead of an OUT token for the transaction. This bit is set at the same time as the USB_EP0_CSRn_H.TXPKTRDY bit is set.                                                                                                                   |                            |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                           | No Request                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                           | Send SETUP token           |

**Table 22-44:** USB\_EP0\_CSRn\_H Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | RXSTALL  | Rx Stall.<br>The USB_EP0_CSRn_H.RXSTALL bit indicates (in host mode) when a STALL handshake is received. The processor core should clear this bit.                                                                                                                                       |
|         |          | 0      No Status                                                                                                                                                                                                                                                                         |
|         |          | 1      Stall Received from Device                                                                                                                                                                                                                                                        |
| 1       | TXPKTRDY | Tx Packet Ready.<br>The USB_EP0_CSRn_H.TXPKTRDY bit should be set (in host mode) by the processor core after loading a data packet into the FIFO. This bit is cleared automatically when the data packet is transmitted. An interrupt is generated (if enabled) when the bit is cleared. |
|         |          | 0      No Tx Packet                                                                                                                                                                                                                                                                      |
|         |          | 1      Tx Packet in Endpoint FIFO                                                                                                                                                                                                                                                        |
| 0       | RXPkTRDY | Rx Packet Ready.<br>The USB_EP0_CSRn_H.RXPkTRDY is set (in host mode) when a data packet is received. An interrupt is generated (if enabled) when this bit is set. The processor core should clear this bit when the packet is read from the FIFO.                                       |
|         |          | 0      No Rx Packet                                                                                                                                                                                                                                                                      |
|         |          | 1      Rx Packet in Endpoint FIFO                                                                                                                                                                                                                                                        |

## EP0 Configuration and Status (Peripheral) Register

The USB\_EP0\_CSRn\_P register provides control and status bits for endpoint 0 in peripheral mode. Note that some bits may be set to clear automatically.

**USB\_EP0\_CSRn\_P: EP0 Configuration and Status (Peripheral) Register - R/W**

Reset = 0x0000

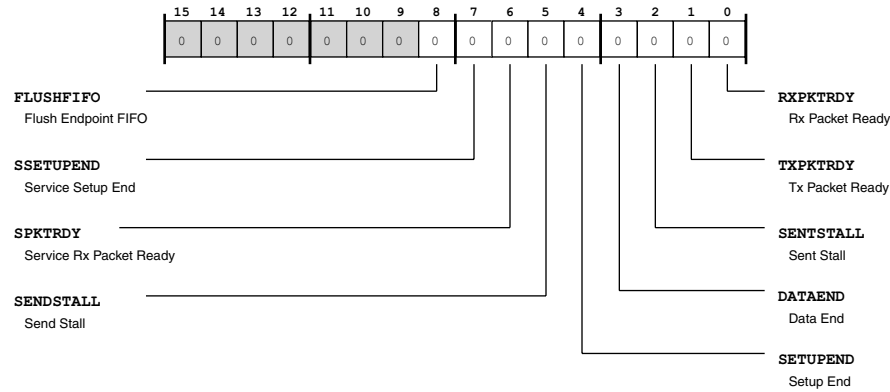


**Figure 22-70:** USB\_EP0\_CSRn\_P Register Diagram

**Table 22-45:** USB\_EP0\_CSRn\_P Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | FLUSHFIFO | Flush Endpoint FIFO.<br>The USB_EP0_CSRn_P.FLUSHFIFO bit directs (in peripheral mode) the USB controller to flush data from the endpoint 0 FIFO and clear the USB_EP0_CSRn_P.TXPKTRDY and USB_EP0_CSRn_P.RXPKTRDY bits. The USB_EP0_CSRn_P.FLUSHFIFO bit should only be set if the USB_EP0_CSRn_P.TXPKTRDY and USB_EP0_CSRn_P.RXPKTRDY bits are set. Note that setting this bit at other times may cause data corruption. |
|         |           | 0 No Flush                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |           | 1 Flush Endpoint FIFO                                                                                                                                                                                                                                                                                                                                                                                                     |
| 7       | SSETUPEND | Service Setup End.<br>The USB_EP0_CSRn_P.SSETUPEND bit is set (in peripheral mode) by the processor core to clear the USB_EP0_CSRn_P.SETUPEND. This bit is cleared automatically.                                                                                                                                                                                                                                         |
|         |           | 0 No Action                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |           | 1 Clear SETUPEND Bit                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 22-45:** USB\_EP0\_CSRn\_P Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | SPKTRDY   | Service Rx Packet Ready.<br>The USB_EP0_CSRn_P.SPKTRDY bit is set (in peripheral mode) by the processor core to clear the USB_EP0_CSRn_P.RXPKTRDY bit. This bit is cleared automatically.                                                                                                                                                                                                                                                                            |
|         |           | 0 No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Clear RXPKTRDY Bit                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 5       | SENDSTALL | Send Stall.<br>The USB_EP0_CSRn_P.SENDSTALL bit is set (in peripheral mode) by the processor core to terminate the current transaction. The STALL handshake is transmitted, then this bit automatically is cleared.                                                                                                                                                                                                                                                  |
|         |           | 0 No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Terminate Current Transaction                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 4       | SETUPEND  | Setup End.<br>The USB_EP0_CSRn_P.SETUPEND bit indicates (in peripheral mode) when a control transaction ends before the USB_EP0_CSRn_P.DATAEND bit is set. An interrupt is generated and the FIFO is flushed at this time. This bit is cleared when the processor core sets the USB_EP0_CSRn_P.SSETUPEND bit.                                                                                                                                                        |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Setup Ended before DATAEND                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 3       | DATAEND   | Data End.<br>The USB_EP0_CSRn_P.DATAEND bit is set (in peripheral mode) by the processor core sets when the core: <ul style="list-style-type: none"> <li>• Sets the USB_EP0_CSRn_P.TXPKTRDY bit for the last data packet.</li> <li>• Clears the USB_EP0_CSRn_P.RXPKTRDY bit after unloading the last data packet.</li> <li>• Sets the USB_EP0_CSRn_P.TXPKTRDY bit for a zero length data packet.</li> </ul> The USB_EP0_CSRn_P.DATAEND bit is cleared automatically. |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Data End Condition                                                                                                                                                                                                                                                                                                                                                                                                                                                 |



**Table 22-45:** USB\_EP0\_CSRn\_P Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                        |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | SENTSTALL | Sent Stall.<br>The USB_EP0_CSRn_P.SENTSTALL bit is set (in peripheral mode) when a STALL handshake is transmitted. The processor core should clear this bit.                                                                                                                                   |
|         |           | 0                      No Status                                                                                                                                                                                                                                                               |
|         |           | 1                      Transmitted STALL Handshake                                                                                                                                                                                                                                             |
| 1       | TXPKTRDY  | Tx Packet Ready.<br>The USB_EP0_CSRn_P.TXPKTRDY bit should be set (in peripheral mode) by the processor core after loading a data packet into the FIFO. This bit is cleared automatically when the data packet is transmitted. An interrupt is generated (if enabled) when the bit is cleared. |
|         |           | 0                                                                                                                                                                                                                                                                                              |
|         |           | 1                      Set this bit after loading a data packet into the FIFO                                                                                                                                                                                                                  |
| 0       | RXPKTRDY  | Rx Packet Ready.<br>The USB_EP0_CSRn_P.RXPKTRDY is set (in peripheral mode) when a data packet is received. An interrupt is generated (if enabled) when this bit is set. The processor core clears this bit by setting the USB_EP0_CSRn_P.SPKTRDY bit.                                         |
|         |           | 0                      No Rx Packet                                                                                                                                                                                                                                                            |
|         |           | 1                      Rx Packet in Endpoint FIFO                                                                                                                                                                                                                                              |

## EPn Transmit Configuration and Status (Host) Register

The USB\_EPn\_TXCSR\_H register provides (in host mode) control and status bits for transfers through the currently selected transmit endpoint.

USB\_EPn\_TXCSR\_H: EPn Transmit Configuration and Status (Host) Register - R/W

Reset = 0x0000

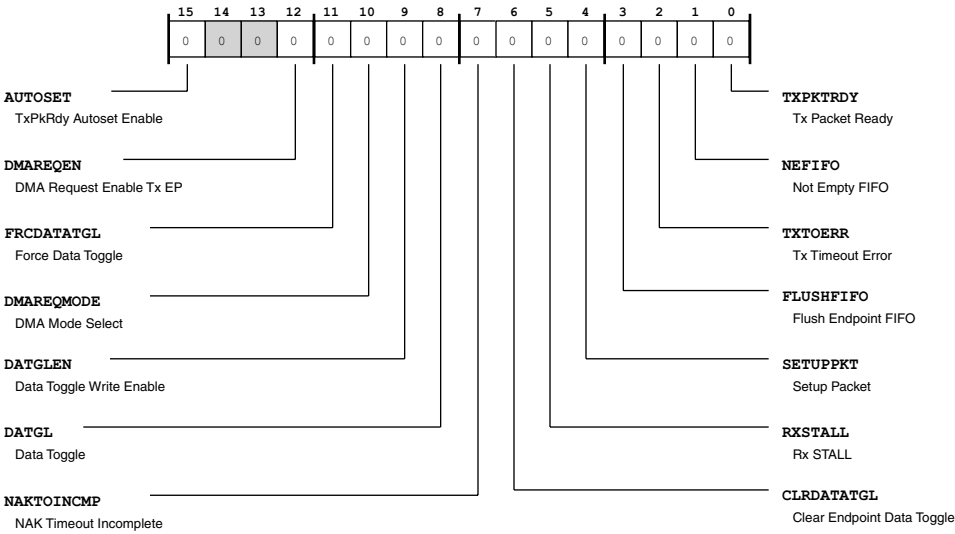


Figure 22-71: USB\_EPn\_TXCSR\_H Register Diagram

Table 22-46: USB\_EPn\_TXCSR\_H Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AUTOSET  | TxPkJdy Autoset Enable.<br>The USB_EPn_TXCSR_H.AUTOSET bit enables (in host mode) automatic setting of the USB_EPn_TXCSR_H.TXPKTRDY bit when the maximum data packet size (USB_EPn_TXMAXP) is loaded into the transmit FIFO. The USB_EPn_TXMAXP value must be a word (4-byte) multiple. If a packet less than the maximum packet size is loaded, the USB_EPn_TXCSR_H.TXPKTRDY bit needs to be set manually. This USB_EPn_TXCSR_H.AUTOSET bit should not be set for high bandwidth endpoints (endpoints with USB_EPn_TXMAXP value greater than 1). |
|         |          | 0      Disable Autoset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Enable Autoset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 22-46:** USB\_EPn\_TXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | DMAREQEN   | DMA Request Enable Tx EP.<br>The USB_EPn_TXCSR_H.DMAREQEN bit enables (in host mode) DMA requests for this transmit endpoint.                                                                                                                                                                                                                                                                                                                                           |
|         |            | 0      Disable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |            | 1      Enable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 11      | FRCDATATGL | Force Data Toggle.<br>The USB_EPn_TXCSR_H.FRCDATATGL bit forces (in host mode) the endpoint data toggle to switch and clears the data packet from the FIFO, regardless of whether an ACK was received. This feature can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.                                                                                                                                   |
|         |            | 0      No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |            | 1      Toggle Endpoint Data                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 10      | DMAREQMODE | DMA Mode Select.<br>The USB_EPn_TXCSR_H.DMAREQMODE bit selects (in host mode) between DMA request mode 1 or 0. This bit must not be cleared the cycle before or the same cycle that the USB_EPn_TXCSR_H.DMAREQEN bit is cleared. In DMA request mode 0, the DMA is programmed to load one packet at a time. Processor intervention is required for each packet. DMA mode 1 can be used with bulk endpoints to transmit multiple packets without processor intervention. |
|         |            | 0      DMA Request Mode 0                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |            | 1      DMA Request Mode 1                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 9       | DATGLEN    | Data Toggle Write Enable.<br>The USB_EPn_TXCSR_H.DATGLEN bit enables (in host mode) the USB controller to write the current state of the endpoint USB_EPn_TXCSR_H.DATGL bit. This bit is automatically cleared once the new value is written.                                                                                                                                                                                                                           |
|         |            | 0      Disable Write to DATGL                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |            | 1      Enable Write to DATGL                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 22-46:** USB\_EPn\_TXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | DATGL      | <p>Data Toggle.</p> <p>The USB_EPn_TXCSR_H.DATGL bit indicates (in host mode) the current state of the endpoint data toggle. If D10 is high, this bit may be written with the required setting of the data toggle. If D10 is low, any value written to this bit is ignored. This bit is only used in host mode.</p>                                                                                                                                                                                 |
|         |            | 0 DATA0 is set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |            | 1 DATA1 is set                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 7       | NAKTOINCMP | <p>NAK Timeout Incomplete.</p> <p>The USB_EPn_TXCSR_H.NAKTOINCMP bit indicates (for bulk endpoints in host mode) when the transmit endpoint is halted following the receipt of NAK responses for longer than the time set in the USB_EPn_TXINTERVAL register. The processor should clear this bit, allowing the endpoint to continue. For high-bandwidth isochronous endpoints in host mode, this bit indicates when no response is received from the device to which the packet is being sent.</p> |
|         |            | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |            | 1 NAK Timeout Over Maximum                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 6       | CLRDATATGL | <p>Clear Endpoint Data Toggle.</p> <p>The USB_EPn_TXCSR_H.CLRDATATGL bit is set (in host mode) by the processor to reset the endpoint data toggle to 0.</p>                                                                                                                                                                                                                                                                                                                                         |
|         |            | 0 No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |            | 1 Reset EP Data Toggle to 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 5       | RXSTALL    | <p>Rx STALL.</p> <p>The USB_EPn_TXCSR_H.RXSTALL bit indicates (in host mode) when a STALL handshake is received. The processor core should clear this bit.</p>                                                                                                                                                                                                                                                                                                                                      |
|         |            | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |            | 1 Stall Received from Device                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Table 22-46:** USB\_EPn\_TXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | SETUPPKT  | Setup Packet.<br>The USB_EPn_TXCSR_H.SETUPPKT bit directs (in host mode) the USB controller to send a SETUP token instead of an OUT token for the transaction. This bit is set at the same time as the USB_EPn_TXCSR_H.TXPKTRDY bit is set.                                                                                                                                                                                                              |
|         |           | 0 No Request                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |           | 1 Send SETUP Token                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 3       | FLUSHFIFO | Flush Endpoint FIFO.<br>The USB_EPn_TXCSR_H.FLUSHFIFO bit directs (in host mode) the USB controller to flush data from the endpoint FIFO and clear the USB_EPn_TXCSR_H.TXPKTRDY bit. The USB_EPn_TXCSR_H.FLUSHFIFO bit should only be set if the USB_EPn_TXCSR_H.TXPKTRDY bit is set. Note that setting this bit at other times may cause data corruption.                                                                                               |
|         |           | 0 No Flush                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |           | 1 Flush endpoint FIFO                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 2       | TXTOERR   | Tx Timeout Error.<br>The USB_EPn_TXCSR_H.TXTOERR bit indicates (in host mode) when three attempts have been made to send a packet and no handshake packet has been received. The USB controller generates an interrupt for this condition, clears the USB_EPn_TXCSR_H.TXPKTRDY bit, and flushes the FIFO. The processor should clear this bit. Note that USB_EPn_TXCSR_H.TXTOERR is valid only when the endpoint is operating in bulk or interrupt mode. |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |           | 1 Tx Timeout Error                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Table 22-46:** USB\_EPn\_TXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | NEFIFO   | Not Empty FIFO.<br>The USB_EPn_TXCSR_H.NEFIFO bit indicates (in host mode) when there is at least one packet in the transmit FIFO. This bit is cleared automatically when a data packet has been transmitted. If the endpoints transmit interrupt is enabled (in USB_INTRTXE), the USB controller generates an interrupt for this condition. Note that the USB_EPn_TXCSR_H.TXPKTRDY bit is also automatically cleared prior to loading a second packet into a double-buffered FIFO. |
|         |          | 0 FIFO Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 FIFO Not Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 0       | TXPKTRDY | Tx Packet Ready.<br>The USB_EPn_TXCSR_H.TXPKTRDY bit should be set (in host mode) by the processor core after loading a data packet into the FIFO. This bit is cleared automatically when the data packet is transmitted. An interrupt is generated (if enabled) when the bit is cleared.                                                                                                                                                                                           |
|         |          | 0 No Tx Packet                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Tx Packet in Endpoint FIFO                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## EPn Transmit Configuration and Status (Peripheral) Register

The USB\_EPn\_TXCSR\_P register provides (in peripheral mode) control and status bits for transfers through the currently selected transmit endpoint.

**USB\_EPn\_TXCSR\_P: EPn Transmit Configuration and Status (Peripheral) Register - R/W**

Reset = 0x0000

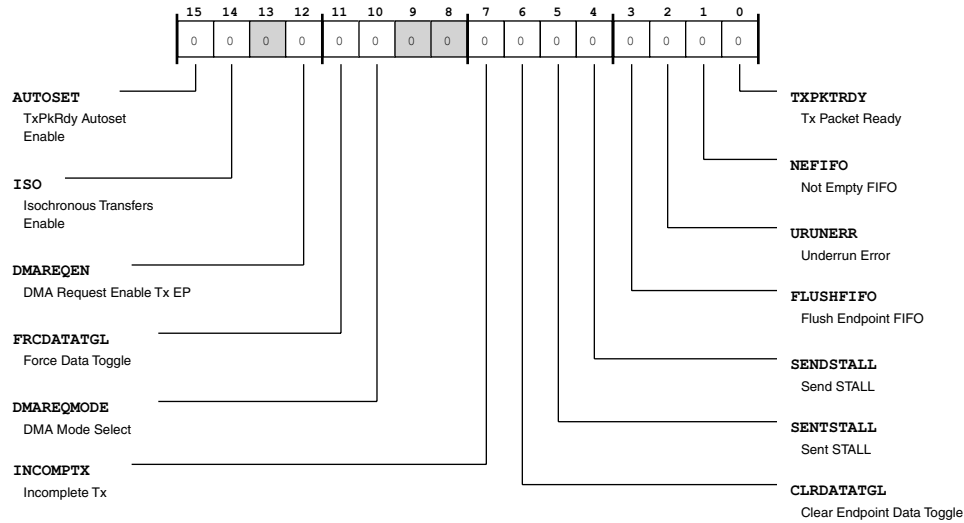


**Figure 22-72:** USB\_EPn\_TXCSR\_P Register Diagram

**Table 22-47:** USB\_EPn\_TXCSR\_P Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AUTOSET  | TxPkrdy Autoset Enable.<br>The USB_EPn_TXCSR_P.AUTOSET bit enables (in peripheral mode) automatic setting of the USB_EPn_TXCSR_P.TXPKTRDY bit when the maximum data packet size (USB_EPn_TXMAXP) is loaded into the transmit FIFO. The USB_EPn_TXMAXP value must be a word (4-byte) multiple. If a packet less than the maximum packet size is loaded, the USB_EPn_TXCSR_P.TXPKTRDY bit needs to be set manually. This USB_EPn_TXCSR_P.AUTOSET bit should not be set for high bandwidth endpoints (endpoints with USB_EPn_TXMAXP value greater than 1). |
|         |          | 0 Disable Autoset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Enable Autoset                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Table 22-47:** USB\_EPn\_TXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | ISO        | Isochronous Transfers Enable.<br>The USB_EPn_TXCSR_P.ISO bit enables (in peripheral mode) the transmit endpoint for isochronous transfers. This bit should be disabled for bulk or interrupt endpoints.                                                                                                                                                                                                                                                                       |
|         |            | 0      Disable Tx EP Isochronous Transfers                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |            | 1      Enable Tx EP Isochronous Transfers                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 12      | DMAREQEN   | DMA Request Enable Tx EP.<br>The USB_EPn_TXCSR_P.DMAREQEN bit enables (in peripheral mode) DMA requests for this transmit endpoint.                                                                                                                                                                                                                                                                                                                                           |
|         |            | 0      Disable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |            | 1      Enable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 11      | FRCDATATGL | Force Data Toggle.<br>The USB_EPn_TXCSR_P.FRCDATATGL bit forces (in peripheral mode) the endpoint data toggle to switch and clears the data packet from the FIFO, regardless of whether an ACK was received. This feature can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.                                                                                                                                   |
|         |            | 0      No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |            | 1      Toggle Endpoint Data                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 10      | DMAREQMODE | DMA Mode Select.<br>The USB_EPn_TXCSR_P.DMAREQMODE bit selects (in peripheral mode) between DMA request mode 1 or 0. This bit must not be cleared the cycle before or the same cycle that the USB_EPn_TXCSR_P.DMAREQEN bit is cleared. In DMA request mode 0, the DMA is programmed to load one packet at a time. Processor intervention is required for each packet. DMA mode 1 can be used with bulk endpoints to transmit multiple packets without processor intervention. |
|         |            | 0      DMA Request Mode 0                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |            | 1      DMA Request Mode 1                                                                                                                                                                                                                                                                                                                                                                                                                                                     |



**Table 22-47:** USB\_EPn\_TXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | INCOMPTX   | Incomplete Tx.<br>The USB_EPn_TXCSR_P.INCOMPTX bit indicates (for high-bandwidth isochronous endpoints in peripheral mode) when a large packet has been split into two or three packets for transmission, but insufficient IN tokens have been received to send all parts.             |
|         |            | 0      No Status                                                                                                                                                                                                                                                                       |
|         |            | 1      Incomplete Tx (Insufficient IN Tokens)                                                                                                                                                                                                                                          |
| 6       | CLRDATATGL | Clear Endpoint Data Toggle.<br>The USB_EPn_TXCSR_P.CLRDATATGL bit is set (in peripheral mode) by the processor to reset the endpoint data toggle to 0.                                                                                                                                 |
|         |            | 0      No Action                                                                                                                                                                                                                                                                       |
|         |            | 1      Reset EP Data Toggle to 0                                                                                                                                                                                                                                                       |
| 5       | SENTSTALL  | Sent STALL.<br>The USB_EPn_TXCSR_P.SENTSTALL bit indicates (in peripheral mode) when the USB controller transmits a STALL handshake. When this condition occurs, the USB controller flushes the FIFO and clears the USB_EPn_TXCSR_P.TXPKTRDY bit. The processor should clear this bit. |
|         |            | 0      No Status                                                                                                                                                                                                                                                                       |
|         |            | 1      STALL Handshake Transmitted                                                                                                                                                                                                                                                     |
| 4       | SENDSTALL  | Send STALL.<br>The USB_EPn_TXCSR_P.SENDSTALL bit (in peripheral mode) is set by the processor to issue a STALL handshake to an IN token. The processor clears this bit to terminate the stall condition. This bit has no effect for isochronous transfers.                             |
|         |            | 0      No Request                                                                                                                                                                                                                                                                      |
|         |            | 1      Request STALL Handshake Transmission                                                                                                                                                                                                                                            |

**Table 22-47:** USB\_EPn\_TXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                     |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 3       | FLUSHFIFO | Flush Endpoint FIFO.<br>The USB_EPn_TXCSR_P.FLUSHFIFO bit directs (in peripheral mode) the USB controller to flush data from the endpoint FIFO and clear the USB_EPn_TXCSR_P.TXPKTRDY bit. The USB_EPn_TXCSR_P.FLUSHFIFO bit should only be set if the USB_EPn_TXCSR_P.TXPKTRDY bit is set. Note that setting this bit at other times may cause data corruption.                                                                                                                          |                     |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Flush            |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Flush endpoint FIFO |
| 2       | URUNERR   | Underrun Error.<br>The USB_EPn_TXCSR_P.URUNERR bit indicates (in peripheral mode) when an IN token is received while the USB_EPn_TXCSR_P.TXPKTRDY bit is not set. The processor should clear this bit.                                                                                                                                                                                                                                                                                    |                     |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Status           |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Underrun Error      |
| 1       | NEFIFO    | Not Empty FIFO.<br>The USB_EPn_TXCSR_P.NEFIFO bit indicates (in peripheral mode) when there is at least one packet in the transmit FIFO. This bit is cleared automatically when a data packet has been transmitted. If the endpoints transmit interrupt is enabled (in USB_INTRTXE), the USB controller generates an interrupt for this condition. Note that the USB_EPn_TXCSR_P.TXPKTRDY bit is also automatically cleared prior to loading a second packet into a double-buffered FIFO. |                     |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | FIFO Empty          |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | FIFO Not Empty      |

**Table 22-47:** USB\_EPn\_TXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | TXPKTRDY | Tx Packet Ready.<br>The USB_EPn_TXCSR_P.TXPKTRDY bit should be set (in peripheral mode) by the processor core after loading a data packet into the FIFO. This bit is cleared automatically when the data packet is transmitted. An interrupt is generated (if enabled) when the bit is cleared. |
|         |          | 0 No Tx Packet                                                                                                                                                                                                                                                                                  |
|         |          | 1 Tx Packet in Endpoint FIFO                                                                                                                                                                                                                                                                    |

## EPn Receive Maximum Packet Length Register

The USB\_EPn\_RXMAXP register defines the maximum amount of data that can be transferred through the selected receive endpoint in a single frame.

Note that a value greater than the maximum allowed of 1023 for full-speed USB operation produces unpredictable results. Also note that the total amount of data represented by the value written to this register must not exceed the receive FIFO size, and should not exceed half the FIFO size if double-buffering is required.

**USB\_EPn\_RXMAXP: EPn Receive Maximum Packet Length Register - R/W**

Reset = 0x0000

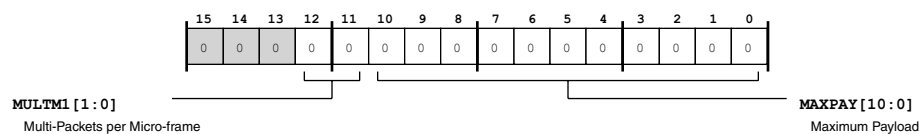


**Figure 22-73:** USB\_EPn\_RXMAXP Register Diagram

**Table 22-48:** USB\_EPn\_RXMAXP Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12:11   | MULTM1   | Multi-Packets per Micro-frame.<br>The USB_EPn_RXMAXP.MULTM1 bits select the number of high-speed high-bandwidth isochronous or interrupt packets that may be transferred in a microframe. The valid number of packets per microframe is 1-3 which corresponds to settings 0-2. If this field is not zero, the USB controller combines multiple packets received within a microframe into a single packet in the FIFO. |

Table 22-48: USB\_EPn\_RXMAXP Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10:0    | MAXPAY   | Maximum Payload.<br>The USB_EPn_RXMAXP.MAXPAY bits select the maximum number of bytes that may be transferred per transaction. This field can be up to 1024 but is subject to constraints by the USB specification based on endpoint mode and speed. This field should not exceed the FIFO size for the endpoint, or half the FIFO size if double buffering is used. This value should match the wMaxPacketSize field of the standard endpoint descriptor (USB 2.0 spec, section 9). The USB_EPn_RXMAXP.MAXPAY bits must be set to an even number of bytes for proper interrupt generation in DMA mode 1. |

EPn Receive Configuration and Status (Host) Register

The USB\_EPn\_RXCSR\_H register provides (in host mode) control and status bits for transfers through the currently selected receive endpoint.

USB\_EPn\_RXCSR\_H: EPn Receive Configuration and Status (Host) Register - R/W

Reset = 0x0000

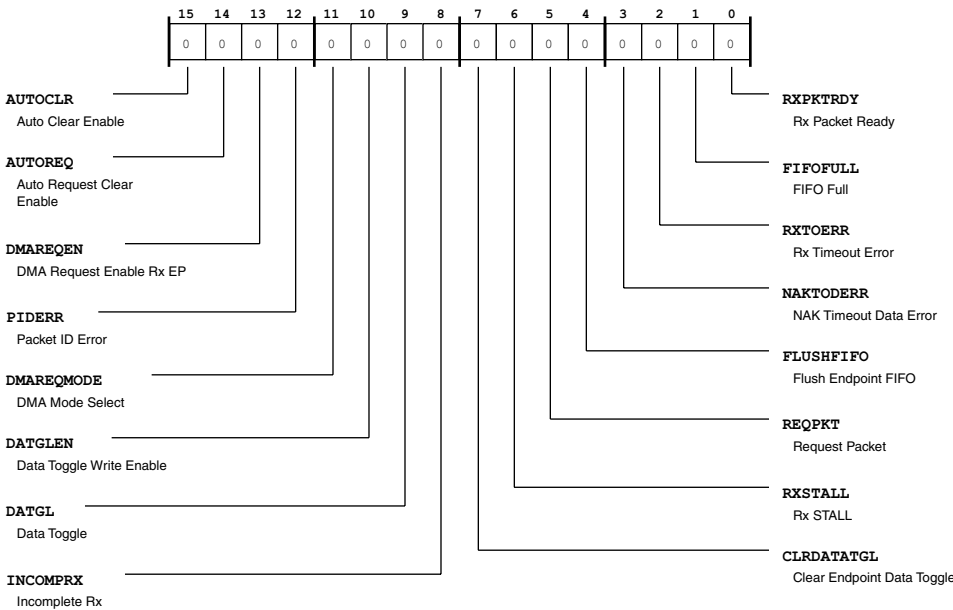


Figure 22-74: USB\_EPn\_RXCSR\_H Register Diagram

**Table 22-49:** USB\_EPn\_RXCSR\_H Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AUTOCLR  | <p>Auto Clear Enable.</p> <p>The USB_EPn_RXCSR_H.AUTOCLR bit directs (in host mode) the USB controller to automatically clear the USB_EPn_RXCSR_H.RXPkTRDY bit when a packet of size USB_EPn_RXMAXP bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, the processor must clear USB_EPn_RXCSR_H.RXPkTRDY manually. When using the DMA to unload the receive FIFO, data is read from the receive FIFO in four byte chunks, regardless of the USB_EPn_RXMAXP value. The USB controller auto clears the USB_EPn_RXCSR_H.RXPkTRDY bit as follows. (In the following: Remainder=(RxMaxP/4), and PktSz-Clearing-RxPktRdy=Actual-Bytes-Read-Packet-Sizes-That-Clear-RxPktRdy.)</p> <ul style="list-style-type: none"> <li>Remainder=0, Bytes-Read=RxMaxP, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1, RxMaxP-2, RxMaxP-3</li> <li>Remainder=3, Bytes Read=RxMaxP+1, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1, RxMaxP-2</li> <li>Remainder=2, Bytes Read=RxMaxP+2, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1</li> <li>Remainder=1, Bytes Read=RxMaxP+3, PktSz-Clearing-RxPktRdy=RxMaxP</li> </ul> <p>Note that the USB_EPn_RXCSR_H.AUTOCLR bit should not be set for high-bandwidth isochronous endpoints.</p> |
|         |          | 0      Disable Auto Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable Auto Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 14      | AUTOREQ  | <p>Auto Request Clear Enable.</p> <p>The USB_EPn_RXCSR_H.AUTOREQ bit directs (in host mode) the USB controller to automatically clear the USB_EPn_RXCSR_H.REQPkt bit when USB_EPn_RXCSR_H.RXPkTRDY bit is cleared. This bit is automatically cleared when a short packet is received.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 0      Disable Auto Request Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1      Enable Auto Request Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

**Table 22-49:** USB\_EPn\_RXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | DMAREQEN   | DMA Request Enable Rx EP.<br>The USB_EPn_RXCSR_H.DMAREQEN bit enables (in host mode) DMA requests for this receive endpoint.                                                                                                                                                                                                                                                                                                                                            |
|         |            | 0      Disable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |            | 1      Enable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 12      | PIDERR     | Packet ID Error.<br>The USB_EPn_RXCSR_H.PIDERR bit indicates (in host mode) when a PID error occurs for isochronous transactions. This bit is ignored in host mode for bulk or interrupt transactions.                                                                                                                                                                                                                                                                  |
|         |            | 0      No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |            | 1      PID Error                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 11      | DMAREQMODE | DMA Mode Select.<br>The USB_EPn_RXCSR_H.DMAREQMODE bit selects (in host mode) between DMA request mode 1 or 0. This bit must not be cleared the cycle before or the same cycle that the USB_EPn_RXCSR_H.DMAREQEN bit is cleared. In DMA request mode 0, the DMA is programmed to load one packet at a time. Processor intervention is required for each packet. DMA mode 1 can be used with bulk endpoints to transmit multiple packets without processor intervention. |
|         |            | 0      DMA Request Mode 0                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |            | 1      DMA Request Mode 1                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 10      | DATGLEN    | Data Toggle Write Enable.<br>The USB_EPn_RXCSR_H.DATGLEN bit enables (in host mode) the USB controller to write the current state of the endpoint USB_EPn_RXCSR_H.DATGL bit. This bit is automatically cleared once the new value is written.                                                                                                                                                                                                                           |
|         |            | 0      Disable Write to DATGL                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |            | 1      Enable Write to DATGL                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 22-49:** USB\_EPn\_RXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                     |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | DATGL      | <p><b>Data Toggle.</b><br/>The USB_EPn_RXCSR_H.DATGL bit indicates (in host mode) the current state of the endpoint data toggle. If D10 is high, this bit may be written with the required setting of the data toggle. If D10 is low, any value written to this bit is ignored. This bit is only used in host mode.</p>                     |
|         |            | 0      DATA0 is Set                                                                                                                                                                                                                                                                                                                         |
|         |            | 1      DATA1 is Set                                                                                                                                                                                                                                                                                                                         |
| 8       | INCOMPRX   | <p><b>Incomplete Rx.</b><br/>The USB_EPn_RXCSR_H.INCOMPRX bit indicates (in host mode for high-bandwidth isochronous or interrupt transfers) when the received packet is incomplete because parts of the packet were not received. This bit is cleared when USB_EPn_RXCSR_H.RXPkTRDY is cleared. For all other modes, this bit is zero.</p> |
|         |            | 0      No Status                                                                                                                                                                                                                                                                                                                            |
|         |            | 1      Incomplete Rx                                                                                                                                                                                                                                                                                                                        |
| 7       | CLRDATATGL | <p><b>Clear Endpoint Data Toggle.</b><br/>The USB_EPn_RXCSR_H.CLRDATATGL bit is set (in host mode) by the processor to reset the endpoint data toggle to 0.</p>                                                                                                                                                                             |
|         |            | 0      No Action                                                                                                                                                                                                                                                                                                                            |
|         |            | 1      Reset EP Data Toggle to 0                                                                                                                                                                                                                                                                                                            |
| 6       | RXSTALL    | <p><b>Rx STALL.</b><br/>The USB_EPn_RXCSR_H.RXSTALL bit indicates (in host mode) when a STALL handshake is received. The processor core should clear this bit.</p>                                                                                                                                                                          |
|         |            | 0      No Status                                                                                                                                                                                                                                                                                                                            |
|         |            | 1      Stall Received from Device                                                                                                                                                                                                                                                                                                           |

**Table 22-49:** USB\_EPn\_RXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | REQPKT    | Request Packet.<br>The USB_EPn_RXCSR_H.REQPKT bit directs (in host mode) the USB controller to request an IN transaction. This bit is cleared when USB_EPn_RXCSR_H.RXPkTRDY is set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |           | 0 No Request                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |           | 1 Send IN Tokens to Device                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 4       | FLUSHFIFO | Flush Endpoint FIFO.<br>The USB_EPn_RXCSR_H.FLUSHFIFO bit directs (in host mode) the USB controller to flush data from the endpoint FIFO and clear the USB_EPn_RXCSR_H.RXPkTRDY bit. The USB_EPn_RXCSR_H.FLUSHFIFO bit should only be set if the USB_EPn_RXCSR_H.RXPkTRDY bit is set. Note that setting this bit at other times may cause data corruption.                                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 0 No Flush                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 1 Flush Endpoint FIFO                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 3       | NAKTODERR | NAK Timeout Data Error.<br>The USB_EPn_RXCSR_H.NAKTODERR bit indicates (in host mode for isochronous transfers) a NAK timeout data error when the USB_EPn_RXCSR_H.RXPkTRDY bit is set and the data packet has a CRC or bit-stuff error. This bit is cleared when the USB_EPn_RXCSR_H.RXPkTRDY bit is cleared.<br>The USB_EPn_RXCSR_H.NAKTODERR bit indicates (in host mod for bulk transfers) when a receive endpoint is halted following the receipt of NAK responses greater than the limit set in the USB_EPn_RXINTERVAL register. The processor should clear this bit to allow the endpoint to continue. If double packet buffering is enabled, the USB_EPn_RXCSR_H.REQPKT bit should also be set in the same cycle as this bit is cleared. |
|         |           | 0 No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |           | 1 NAK Timeout Data Error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



**Table 22-49:** USB\_EPn\_RXCSR\_H Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                 |                            |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| 2       | RXTOERR  | Rx Timeout Error.<br>The USB_EPn_RXCSR_H.RXTOERR bit indicates (in host mode) when three attempts have been made to receive a packet and no data packet has been received. The USB controller generates an interrupt for this condition. The processor should clear this bit. Note that USB_EPn_RXCSR_H.RXTOERR is valid only when the endpoint is operating in bulk or interrupt mode. |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                       | No Status                  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                       | Rx Timeout Error           |
| 1       | FIFOFULL | FIFO Full.<br>The USB_EPn_RXCSR_H.FIFOFULL bit indicates (in host mode) when no more packets can be loaded into the receive FIFO.                                                                                                                                                                                                                                                       |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                       | No Status                  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                       | FIFO Full                  |
| 0       | RXPkTRDY | Rx Packet Ready.<br>The USB_EPn_RXCSR_H.RXPkTRDY is set (in host mode) when a data packet is received. An interrupt is generated (if enabled) when this bit is set. The processor core should clear this bit when the packet is read from the FIFO.                                                                                                                                     |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                       | No Rx Packet               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                       | Rx Packet in Endpoint FIFO |

## EPn Receive Configuration and Status (Peripheral) Register

The USB\_EPn\_RXCSR\_P register provides (in peripheral mode) control and status bits for transfers through the currently selected receive endpoint.

USB\_EPn\_RXCSR\_P: EPn Receive Configuration and Status (Peripheral) Register - R/W

Reset = 0x0000

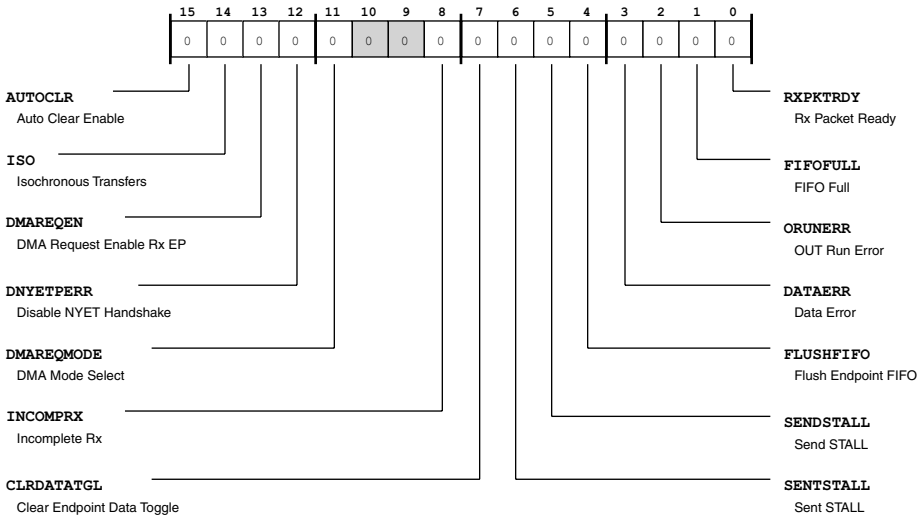


Figure 22-75: USB\_EPn\_RXCSR\_P Register Diagram

**Table 22-50:** USB\_EPn\_RXCSR\_P Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AUTOCLR  | <p>Auto Clear Enable.</p> <p>The USB_EPn_RXCSR_P.AUTOCLR bit directs (in peripheral mode) the USB controller to automatically clear the USB_EPn_RXCSR_P.RXPkTRDY bit when a packet of size USB_EPn_RXMAXP bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, the processor must clear USB_EPn_RXCSR_P.RXPkTRDY manually. When using the DMA to unload the receive FIFO, data is read from the receive FIFO in four byte chunks, regardless of the USB_EPn_RXMAXP value. The USB controller auto clears the USB_EPn_RXCSR_P.RXPkTRDY bit as follows. (In the following: Remainder=(RxMaxP/4), and PktSz-Clearing-RxPktRdy=Actual-Bytes-Read-Packet-Sizes-That-Clear-RxPktRdy.)</p> <ul style="list-style-type: none"> <li>• Remainder=0, Bytes-Read=RxMaxP, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1, RxMaxP-2, RxMaxP-3</li> <li>• Remainder=3, Bytes Read=RxMaxP+1, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1, RxMaxP-2</li> <li>• Remainder=2, Bytes Read=RxMaxP+2, PktSz-Clearing-RxPktRdy=RxMaxP, RxMaxP-1</li> <li>• Remainder=1, Bytes Read=RxMaxP+3, PktSz-Clearing-RxPktRdy=RxMaxP</li> </ul> <p>Note that the USB_EPn_RXCSR_P.AUTOCLR bit should not be set for high-bandwidth isochronous endpoints.</p> |
|         |          | 0      Disable Auto Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable Auto Clear                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 14      | ISO      | <p>Isochronous Transfers.</p> <p>The USB_EPn_RXCSR_P.ISO bit selects (in peripheral mode) between isochronous transfers and bulk/interrupt transfers.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0      This bit should be cleared for bulk or interrupt transfers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      This bit should be set for isochronous transfers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 22-50:** USB\_EPn\_RXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | DMAREQEN   | DMA Request Enable Rx EP.<br>The USB_EPn_RXCSR_P.DMAREQEN bit enables (in peripheral mode) DMA requests for this receive endpoint.                                                                                                                                                                                                                                                                                                                                            |
|         |            | 0      Disable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |            | 1      Enable DMA Request                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 12      | DNYETPERR  | Disable NYET Handshake.<br>The USB_EPn_RXCSR_P.DNYETPERR bit disables (in peripheral mode for high speed bulk/interrupt transactions) NYET handshakes. When this bit is set, all successful receive packets are ACK'd, including the point at which the FIFO becomes full. The USB_EPn_RXCSR_P.DNYETPERR bit must be set for all interrupt endpoints in high speed mode.                                                                                                      |
|         |            | 0      Enable NYET Handshake                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |            | 1      Disable NYET Handshake                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 11      | DMAREQMODE | DMA Mode Select.<br>The USB_EPn_RXCSR_P.DMAREQMODE bit selects (in peripheral mode) between DMA request mode 1 or 0. This bit must not be cleared the cycle before or the same cycle that the USB_EPn_RXCSR_P.DMAREQEN bit is cleared. In DMA request mode 0, the DMA is programmed to load one packet at a time. Processor intervention is required for each packet. DMA mode 1 can be used with bulk endpoints to transmit multiple packets without processor intervention. |
|         |            | 0      DMA Request Mode 0                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |            | 1      DMA Request Mode 1                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 8       | INCOMPRX   | Incomplete Rx.<br>The USB_EPn_RXCSR_P.INCOMPRX bit indicates (in peripheral mode for high-bandwidth isochronous or interrupt transfers) when the received packet is incomplete because parts of the packet were not received. This bit is cleared when USB_EPn_RXCSR_P.RXPKTRDY is cleared. For all other modes, this bit is zero.                                                                                                                                            |
|         |            | 0      No Status                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |            | 1      Incomplete Rx                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

**Table 22-50:** USB\_EPn\_RXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                          |
|---------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | CLRDATATGL | Clear Endpoint Data Toggle.<br>The USB_EPn_RXCSR_P.CLRDATATGL bit is set (in peripheral mode) by the processor to reset the endpoint data toggle to 0.                                                                                                                                                                                                           |
|         |            | 0                      No Action                                                                                                                                                                                                                                                                                                                                 |
|         |            | 1                      Reset EP Data Toggle to 0                                                                                                                                                                                                                                                                                                                 |
| 6       | SENTSTALL  | Sent STALL.<br>The USB_EPn_RXCSR_P.SENTSTALL bit indicates (in peripheral mode) when a STALL handshake is transmitted. The processor should clear this bit.                                                                                                                                                                                                      |
|         |            | 0                      No Status                                                                                                                                                                                                                                                                                                                                 |
|         |            | 1                      STALL Handshake Transmitted                                                                                                                                                                                                                                                                                                               |
| 5       | SENDSTALL  | Send STALL.<br>The USB_EPn_RXCSR_P.SENDSTALL bit is set (in peripheral mode) by the processor to send a STALL handshake. The processor clears this bit to terminate the stall condition. This bit has no effect for isochronous transfers.                                                                                                                       |
|         |            | 0                      No Action                                                                                                                                                                                                                                                                                                                                 |
|         |            | 1                      Request STALL Handshake                                                                                                                                                                                                                                                                                                                   |
| 4       | FLUSHFIFO  | Flush Endpoint FIFO.<br>The USB_EPn_RXCSR_P.FLUSHFIFO bit directs (in peripheral mode) the USB controller to flush data from the endpoint FIFO and clear the USB_EPn_RXCSR_P.RXPKTRDY bit. The USB_EPn_RXCSR_P.FLUSHFIFO bit should only be set if the USB_EPn_RXCSR_P.RXPKTRDY bit is set. Note that setting this bit at other times may cause data corruption. |
|         |            | 0                      No Flush                                                                                                                                                                                                                                                                                                                                  |
|         |            | 1                      Flush Endpoint FIFO                                                                                                                                                                                                                                                                                                                       |

**Table 22-50:** USB\_EPn\_RXCSR\_P Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                    |                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| 3       | DATAERR  | Data Error.<br>The USB_EPn_RXCSR_P.DATAERR bit indicates (in peripheral mode for isochronous transfers) when the USB_EPn_RXCSR_P.RXPKTRDY bit is set and the data packet has a CRC or bit-stuff error. This bit is cleared when USB_EPn_RXCSR_P.RXPKTRDY is cleared. The USB_EPn_RXCSR_P.DATAERR bit is always zero for bulk endpoints in peripheral mode. |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                          | No Status                  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                          | Data Error                 |
| 2       | ORUNERR  | OUT Run Error.<br>The USB_EPn_RXCSR_P.ORUNERR bit indicates (in peripheral mode for isochronous transfers) when an OUT packet cannot be loaded into the receive FIFO. The processor should clear this bit. The USB_EPn_RXCSR_P.ORUNERR bit always returns zero in bulk mode.                                                                               |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                          | No Status                  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                          | OUT Run Error              |
| 1       | FIFOFULL | FIFO Full.<br>The USB_EPn_RXCSR_P.FIFOFULL bit indicates (in peripheral mode) when no more packets can be loaded into the receive FIFO.                                                                                                                                                                                                                    |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                          | No Status                  |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                          | FIFO Full                  |
| 0       | RXPKTRDY | Rx Packet Ready.<br>The USB_EPn_RXCSR_P.RXPKTRDY is set (in peripheral mode) when a data packet is received. An interrupt is generated (if enabled) when this bit is set. The processor core should clear this bit when the packet is read from the FIFO.                                                                                                  |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                          | No Rx Packet               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                          | Rx Packet in Endpoint FIFO |

## EP0 Number of Received Bytes Register

The USB\_EP0\_CNTn register indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while the USB\_EP0\_CSRn\_H.RXPKT RDY bit or USB\_EP0\_CSRn\_P.RXPKT RDY bit is set.

USB\_EP0\_CNTn: EP0 Number of Received Bytes Register - R/NW

Reset = 0x0000

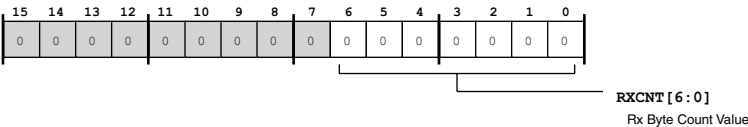


Figure 22-76: USB\_EP0\_CNTn Register Diagram

Table 22-51: USB\_EP0\_CNTn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:0     | RXCNT    | Rx Byte Count Value.<br>The USB_EP0_CNTn.RXCNT bits holds the number of data bytes currently inline ready to be read from the Rx FIFO. The value returned changes as the FIFO is unloaded and is only valid while USB_EP0_CSRn_H.RXPKT RDY bit or USB_EP0_CSRn_P.RXPKT RDY bit is set. |

## EPn Number of Bytes Received Register

The USB\_EPn\_RXCNT register indicates the number of received data bytes in the endpoint receive FIFO. The value returned changes as the contents of the FIFO change and is only valid while the USB\_EPn\_RXCSR\_H.RXPKT RDY bit or USB\_EPn\_RXCSR\_P.RXPKT RDY bit is set.

USB\_EPn\_RXCNT: EPn Number of Bytes Received Register - R/NW

Reset = 0x0000

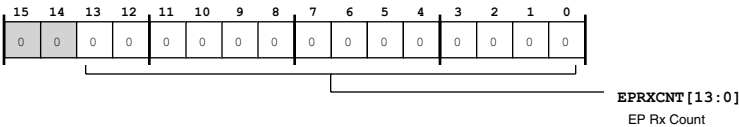


Figure 22-77: USB\_EPn\_RXCNT Register Diagram

Table 22-52: USB\_EPn\_RXCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------|
| 13:0    | EPRXCNT  | EP Rx Count.<br>The USB_EPn_RXCNT.EPRXCNT bits hold the number of data bytes ready to be read from the receive FIFO. |

EPn Transmit Type Register

The USB\_EPn\_TXTYPE register selects the endpoint number and transaction protocol to use for the currently selected transmit endpoint. There is a USB\_EPn\_TXTYPE register for each transmit endpoint. Note that this register is only used in host mode.

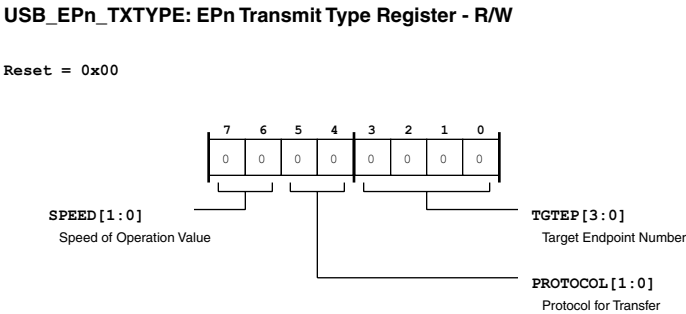


Figure 22-78: USB\_EPn\_TXTYPE Register Diagram



**Table 22-53:** USB\_EPn\_TXTYPE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:6     | SPEED    | Speed of Operation Value.<br>The USB_EPn_TXTYPE.SPEED bits select the USB controller operating speed for the endpoint when acting as a host connected to devices through a hub. In these instances, the USB controller must issue split transactions under certain conditions. If a device is directly connected (not through a hub), all endpoints use the same speed as which the controller is connected. When not connected to devices through a hub, program this field with 00. |
|         |          | 0 Same Speed as the Core                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 High Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 Full Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 3 Low Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 5:4     | PROTOCOL | Protocol for Transfer.<br>The USB_EPn_TXTYPE.PROTOCOL bits select the transfer protocol for the endpoint.                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0 Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Isochronous                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 2 Bulk                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 Interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Table 22-53:** USB\_EPn\_TXTYPE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | TGTEP    | Target Endpoint Number.<br>The USB_EPn_TXTYPE.TGTEP bits select (for endpoints 1-11) the target endpoint. This value should be set to the endpoint number contained in the transmit endpoint descriptor returned during device enumeration. Endpoint 0 always uses target endpoint number 0. (Enumeration values not shown are reserved.) |
|         |          | 0 Endpoint 0                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Endpoint 1                                                                                                                                                                                                                                                                                                                              |
|         |          | 2 Endpoint 2                                                                                                                                                                                                                                                                                                                              |
|         |          | 3 Endpoint 3                                                                                                                                                                                                                                                                                                                              |
|         |          | 4 Endpoint 4                                                                                                                                                                                                                                                                                                                              |
|         |          | 5 Endpoint 5                                                                                                                                                                                                                                                                                                                              |
|         |          | 6 Endpoint 6                                                                                                                                                                                                                                                                                                                              |
|         |          | 7 Endpoint 7                                                                                                                                                                                                                                                                                                                              |
|         |          | 8 Endpoint 8                                                                                                                                                                                                                                                                                                                              |
|         |          | 9 Endpoint 9                                                                                                                                                                                                                                                                                                                              |
|         |          | 10 Endpoint 10                                                                                                                                                                                                                                                                                                                            |
|         |          | 11 Endpoint 11                                                                                                                                                                                                                                                                                                                            |
|         |          | 12 Endpoint 12                                                                                                                                                                                                                                                                                                                            |
|         |          | 13 Endpoint 13                                                                                                                                                                                                                                                                                                                            |
|         |          | 14 Endpoint 14                                                                                                                                                                                                                                                                                                                            |
|         |          | 15 Endpoint 15                                                                                                                                                                                                                                                                                                                            |

## EP0 Connection Type Register

The USB\_EP0\_TYPE<sub>n</sub> register selects the USB controller operating speed for endpoint 0 when acting as a host connected to devices through a hub.

USB\_EP0\_TYPEn: EP0 Connection Type Register - R/W

Reset = 0x00

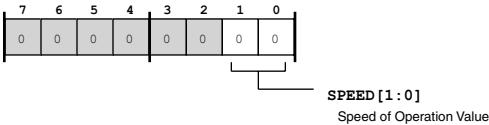


Figure 22-79: USB\_EP0\_TYPEn Register Diagram

Table 22-54: USB\_EP0\_TYPEn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1:0     | SPEED    | Speed of Operation Value.<br>The USB_EP0_TYPEn.SPEED bits select the USB controller operating speed for endpoint 0 when acting as a host connected to devices through a hub. In these instances, the USB controller must issue split transactions under certain conditions. If a device is directly connected (not through a hub), all endpoints use the same speed as which the controller is connected. When not connected to devices through a hub, program this field with 00. |
|         |          | 0 Same Speed as Processor Core                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 High Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 2 Full Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 3 Low Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

EP0 NAK Limit Register

The USB\_EP0\_NAKLIMITn register determines the number of frames/micro-frames after which endpoint 0 should timeout on receiving a stream of NAK responses for bulk endpoints.

USB\_EP0\_NAKLIMITn: EP0 NAK Limit Register - R/W

Reset = 0x00

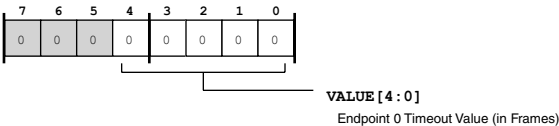


Figure 22-80: USB\_EP0\_NAKLIMITn Register Diagram

**Table 22-55:** USB\_EP0\_NAKLIMITn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------|
| 4:0     | VALUE    | Endpoint 0 Timeout Value (in Frames).<br>The USB_EP0_NAKLIMITn.VALUE bits hold the endpoint 0 timeout value (number of frames). |

**EPn Transmit Polling Interval Register**

The USB\_EPn\_TXINTERVAL register defines the polling interval for the currently selected transmit endpoint for interrupt, isochronous, and bulk transfers. There is a USB\_EPn\_TXINTERVAL register for each configured transmit endpoint, except endpoint 0. The transfer types relate to speed, interval value, and interval operation as follows:

- Interrupt: Speed=Low Speed or Full Speed, USB\_EPn\_TXINTERVAL=1-255, and Operation=Polling interval is m frames.
- Interrupt: Speed=High Speed, USB\_EPn\_TXINTERVAL=1-16, and Operation=Polling interval is 2<sup>(m-1)</sup> micro-frames.
- Isochronous: Speed=Full Speed or High Speed, USB\_EPn\_TXINTERVAL=1-16, and Operation=Polling interval is 2<sup>(m-1)</sup> frames or micro-frames.
- Bulk: Speed=Full Speed or High Speed, USB\_EPn\_TXINTERVAL=2-16, and Operation=NAK Limit is 2<sup>(m-1)</sup> frames or micro-frames.

Note that a USB\_EPn\_TXINTERVAL value of 0 or 1 disables the NAK timeout function.

**USB\_EPn\_TXINTERVAL: EPn Transmit Polling Interval Register - R/W**

Reset = 0x00

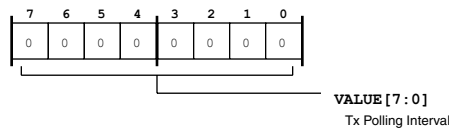


**Figure 22-81:** USB\_EPn\_TXINTERVAL Register Diagram

**Table 22-56:** USB\_EPn\_TXINTERVAL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | <p>Tx Polling Interval.</p> <p>The <code>USB_EPn_TXINTERVAL.VALUE</code> bits define the polling interval value for interrupt and isochronous transfers and select the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses for bulk and control endpoints. Note that the USB controller halts transfers to control endpoints if the host receives NAK responses for more frames than the limit set by this register.</p> |

## EPn Receive Type Register

The `USB_EPn_RXTYPE` register selects the endpoint number and transaction protocol to use for the currently selected receive endpoint. There is a `USB_EPn_RXTYPE` register for each receive endpoint. Note that this register is only used in host mode.

### USB\_EPn\_RXTYPE: EPn Receive Type Register - R/W

Reset = 0x00

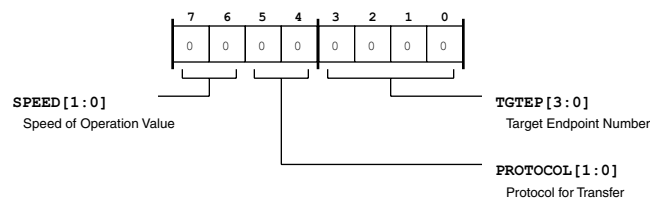


**Figure 22-82:** USB\_EPn\_RXTYPE Register Diagram

**Table 22-57:** USB\_EPn\_RXTYPE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:6     | SPEED    | Speed of Operation Value.<br>The USB_EPn_RXTYPE.SPEED bits select the USB controller operating speed for the endpoint when acting as a host connected to devices through a hub. In these instances, the USB controller must issue split transactions under certain conditions. If a device is directly connected (not through a hub), all endpoints use the same speed as which the controller is connected. When not connected to devices through a hub, program this field with 00. |
|         |          | 0 Same Speed as the Core                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 High Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 Full Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 3 Low Speed                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 5:4     | PROTOCOL | Protocol for Transfer.<br>The USB_EPn_RXTYPE.PROTOCOL bits select the transfer protocol for the endpoint.                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0 Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Isochronous                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 2 Bulk                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 Interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

**Table 22-57:** USB\_EPn\_RXTYPE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | TGTEP    | Target Endpoint Number.<br>The USB_EPn_RXTYPE.TGTEP bits select (for endpoints 1-11) the target endpoint. This value should be set to the endpoint number contained in the receive endpoint descriptor returned during device enumeration. Endpoint 0 always uses target endpoint number 0. (Enumeration values not shown are reserved.) |
|         |          | 0 Endpoint 0                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Endpoint 1                                                                                                                                                                                                                                                                                                                             |
|         |          | 2 Endpoint 2                                                                                                                                                                                                                                                                                                                             |
|         |          | 3 Endpoint 3                                                                                                                                                                                                                                                                                                                             |
|         |          | 4 Endpoint 4                                                                                                                                                                                                                                                                                                                             |
|         |          | 5 Endpoint 5                                                                                                                                                                                                                                                                                                                             |
|         |          | 6 Endpoint 6                                                                                                                                                                                                                                                                                                                             |
|         |          | 7 Endpoint 7                                                                                                                                                                                                                                                                                                                             |
|         |          | 8 Endpoint 8                                                                                                                                                                                                                                                                                                                             |
|         |          | 9 Endpoint 9                                                                                                                                                                                                                                                                                                                             |
|         |          | 10 Endpoint 10                                                                                                                                                                                                                                                                                                                           |
|         |          | 11 Endpoint 11                                                                                                                                                                                                                                                                                                                           |
|         |          | 12 Endpoint 12                                                                                                                                                                                                                                                                                                                           |
|         |          | 13 Endpoint 13                                                                                                                                                                                                                                                                                                                           |
|         |          | 14 Endpoint 14                                                                                                                                                                                                                                                                                                                           |
|         |          | 15 Endpoint 15                                                                                                                                                                                                                                                                                                                           |

## EPn Receive Polling Interval Register

The USB\_EPn\_RXINTERVAL register defines the polling interval for the currently selected receive endpoint for interrupt, isochronous, and bulk transfers. There is a USB\_EPn\_RXINTERVAL register for each config-

ured receive endpoint, except endpoint 0. The transfer types relate to speed, interval value, and interval operation as follows:

- Interrupt: Speed=Low Speed or Full Speed, USB\_EPn\_RXINTERVAL=1-255, and Operation=Polling interval is m frames.
- Interrupt: Speed=High Speed, USB\_EPn\_RXINTERVAL=1-16, and Operation=Polling interval is  $2^{(m-1)}$  micro-frames.
- Isochronous: Speed=Full Speed or High Speed, USB\_EPn\_RXINTERVAL=1-16, and Operation=Polling interval is  $2^{(m-1)}$  frames or micro-frames.
- Bulk: Speed=Full Speed or High Speed, USB\_EPn\_RXINTERVAL=2-16, and Operation=NAK Limit is  $2^{(m-1)}$  frames or micro-frames.

Note that a USB\_EPn\_RXINTERVAL value of 0 or 1 disables the NAK timeout function.

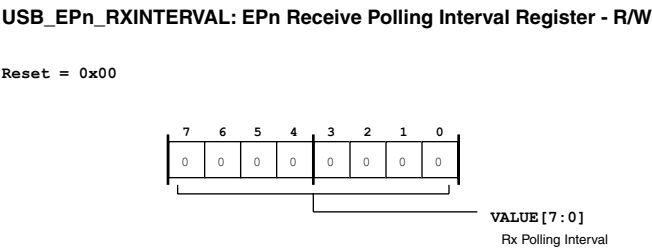


Figure 22-83: USB\_EPn\_RXINTERVAL Register Diagram

Table 22-58: USB\_EPn\_RXINTERVAL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | Rx Polling Interval.<br>The USB_EPn_RXINTERVAL.VALUE bits define the polling interval value for interrupt and isochronous transfers and select the number of frames/microframes after which the endpoint should timeout on receiving a stream of NAK responses for bulk and control endpoints. Note that the USB controller halts transfers to control endpoints if the host receives NAK responses for more frames than the limit set by this register. |

EP0 Configuration Information Register

The USB\_EP0\_CFGDATA<sub>n</sub> register describes the USB controller hardware configuration. This register only exists for endpoint 0.



**USB\_EP0\_CFGDATAn: EP0 Configuration Information Register - R/NW**

Reset = 0x1e

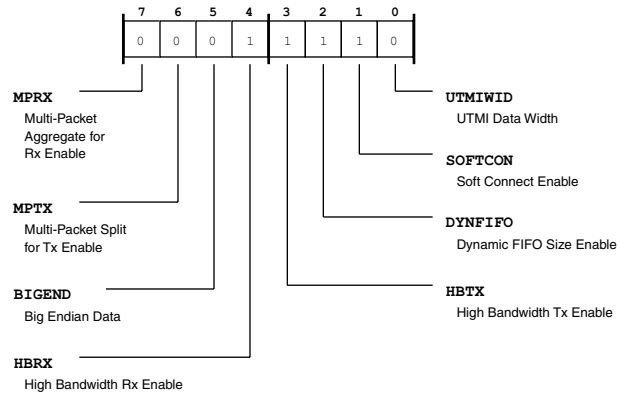


**Figure 22-84:** USB\_EP0\_CFGDATAn Register Diagram

**Table 22-59:** USB\_EP0\_CFGDATAn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | MPRX     | Multi-Packet Aggregate for Rx Enable.<br>The USB_EP0_CFGDATAn.MPRX bit indicates whether the USB controller aggregates receive packets into bulk packets before the processor core reads the data.                                                                                 |
|         |          | 0 No Aggregate Rx Bulk Packets                                                                                                                                                                                                                                                     |
|         |          | 1 Aggregate Rx Bulk Packets                                                                                                                                                                                                                                                        |
| 6       | MPTX     | Multi-Packet Split for Tx Enable.<br>The USB_EP0_CFGDATAn.MPTX bit indicates whether the USB controller permits transmit of large packets through writing to bulk endpoints. The USB controller splits the transmit data into packets, which are appropriately sized for transmit. |
|         |          | 0 No Split Tx Bulk Packets                                                                                                                                                                                                                                                         |
|         |          | 1 Split Tx Bulk Packets                                                                                                                                                                                                                                                            |

**Table 22-59:** USB\_EP0\_CFGDATAn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | BIGEND   | Big Endian Data.<br>The <code>USB_EP0_CFGDATAn.BIGEND</code> bit indicates whether the USB controller uses big endian configuration or little endian configuration.                                                                                                                                                                                                                                                                                  |
|         |          | 0 Little Endian Configuration                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 Big Endian Configuration                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 4       | HBRX     | High Bandwidth Rx Enable.<br>The <code>USB_EP0_CFGDATAn.HBRX</code> bit indicates whether the USB controller supports high-bandwidth receive ISO endpoint support.                                                                                                                                                                                                                                                                                   |
|         |          | 0 No High Bandwidth Rx                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 High Bandwidth Rx                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 3       | HBTX     | High Bandwidth Tx Enable.<br>The <code>USB_EP0_CFGDATAn.HBTX</code> bit indicates whether the USB controller supports high bandwidth transmit ISO endpoint support.                                                                                                                                                                                                                                                                                  |
|         |          | 0 No High Bandwidth Tx                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 High Bandwidth Tx                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 2       | DYNFIFO  | Dynamic FIFO Size Enable.<br>The <code>USB_EP0_CFGDATAn.DYNFIFO</code> bit indicates whether the USB controller uses dynamic FIFO size support, enabling the dynamic FIFO registers, which are accessed using the configuration set in the endpoint's indexed FIFO size and FIFO address registers ( <code>USB_TXFIFOSZ</code> , <code>USB_RXFIFOSZ</code> , <code>USB_TXFIFOADDR</code> , and <code>USB_RXFIFOADDR</code> ), except for endpoint 0. |
|         |          | 0 No Dynamic FIFO Size                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 Dynamic FIFO Size                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Table 22-59:** USB\_EP0\_CFGDATAn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 1       | SOFTCON  | Soft Connect Enable.<br>The USB_EP0_CFGDATAn.SOFTCON bit indicates whether the USB controller uses soft connect.                   |
|         |          | 0 No Soft Connect                                                                                                                  |
|         |          | 1 Soft Connect                                                                                                                     |
| 0       | UTMIWID  | UTMI Data Width.<br>The USB_EP0_CFGDATAn.UTMIWID bit indicates whether the USB controller uses an 8-bit or 16-bit UTMI data width. |
|         |          | 0 8-bit UTMI Data Width                                                                                                            |
|         |          | 1 16-bit UTMI Data Width                                                                                                           |

## DMA Interrupt Register

The USB\_DMA\_IRQ register indicates which of the DMA master channels have a pending interrupt. The USB controller generates the interrupt when the corresponding DMA count register (USB\_DMA<sub>n</sub>\_CNT) reaches zero. The USB controller clears this register when it is read.

**USB\_DMA\_IRQ: DMA Interrupt Register - R/NW**

Reset = 0x00

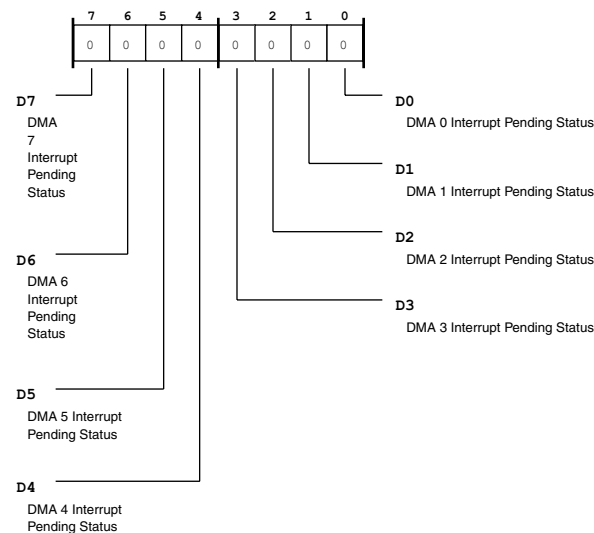


**Figure 22-85:** USB\_DMA\_IRQ Register Diagram

**Table 22-60:** USB\_DMA\_IRQ Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 7       | D7       | DMA 7 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D7</code> indicates whether there is a DMA 7 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 6       | D6       | DMA 6 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D6</code> indicates whether there is a DMA 6 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 5       | D5       | DMA 5 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D5</code> indicates whether there is a DMA 5 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 4       | D4       | DMA 4 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D4</code> indicates whether there is a DMA 4 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 3       | D3       | DMA 3 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D3</code> indicates whether there is a DMA 3 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |

**Table 22-60:** USB\_DMA\_IRQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 2       | D2       | DMA 2 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D2</code> indicates whether there is a DMA 2 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 1       | D1       | DMA 1 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D1</code> indicates whether there is a DMA 1 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |
| 0       | D0       | DMA 0 Interrupt Pending Status.<br>The <code>USB_DMA_IRQ.D0</code> indicates whether there is a DMA 0 interrupt pending. |
|         |          | 0      No Pending Interrupt                                                                                              |
|         |          | 1      Pending DMA Interrupt                                                                                             |

## DMA Channel n Control Register

There is a `USB_DMAn_CTL` register for each DMA master channel. This register assigns, configures, and controls each endpoint with a corresponding DMA master channel.

USB\_DMAn\_CTL: DMA Channel n Control Register - R/W

Reset = 0x0000

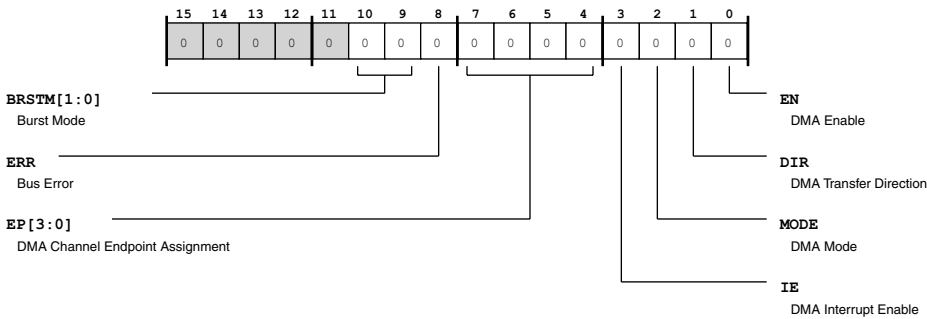


Figure 22-86: USB\_DMAn\_CTL Register Diagram

Table 22-61: USB\_DMAn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10:9    | BRSTM    | Burst Mode.<br>The USB_DMAn_CTL . BRSTM bits select the type or length of burst transfer used by the corresponding DMA channel to transfer data.            |
|         |          | 0 Unspecified Length                                                                                                                                        |
|         |          | 1 INCR4 or Unspecified Length                                                                                                                               |
|         |          | 2 INCR8, INCR4, or Unspecified Length                                                                                                                       |
|         |          | 3 INCR16, INCR8, INCR4, or Unspecified Length                                                                                                               |
| 8       | ERR      | Bus Error.<br>The USB_DMAn_CTL . ERR bit indicates when a peripheral bus error has been encountered by the master channel. This bit is cleared by software. |
|         |          | 0 No Status                                                                                                                                                 |
|         |          | 1 Bus Error                                                                                                                                                 |

**Table 22-61:** USB\_DMA<sub>n</sub>\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | EP       | DMA Channel Endpoint Assignment.<br>The USB_DMA <sub>n</sub> _CTL . EP bits select the endpoint assignments for the DMA channel. (Enumeration values not shown are reserved.)                  |
|         |          | 0 Endpoint 0                                                                                                                                                                                   |
|         |          | 1 Endpoint 1                                                                                                                                                                                   |
|         |          | 2 Endpoint 2                                                                                                                                                                                   |
|         |          | 3 Endpoint 3                                                                                                                                                                                   |
|         |          | 4 Endpoint 4                                                                                                                                                                                   |
|         |          | 5 Endpoint 5                                                                                                                                                                                   |
|         |          | 6 Endpoint 6                                                                                                                                                                                   |
|         |          | 7 Endpoint 7                                                                                                                                                                                   |
|         |          | 8 Endpoint 8                                                                                                                                                                                   |
|         |          | 9 Endpoint 9                                                                                                                                                                                   |
|         |          | 10 Endpoint 10                                                                                                                                                                                 |
|         |          | 11 Endpoint 11                                                                                                                                                                                 |
|         |          | 12 Endpoint 12                                                                                                                                                                                 |
|         |          | 13 Endpoint 13                                                                                                                                                                                 |
|         |          | 14 Endpoint 14                                                                                                                                                                                 |
|         |          | 15 Endpoint 15                                                                                                                                                                                 |
| 3       | IE       | DMA Interrupt Enable.<br>The USB_DMA <sub>n</sub> _CTL . IE bit enables DMA interrupts for the DMA channel, enabling operation of the channel's corresponding bit in the USB_DMA_IRQ register. |
|         |          | 0 Disable Interrupt                                                                                                                                                                            |
|         |          | 1 Enable Interrupt                                                                                                                                                                             |

**Table 22-61:** USB\_DMA<sub>n</sub>\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | MODE     | DMA Mode.<br>The USB_DMA <sub>n</sub> _CTL.MODE bit selects whether the DMA channel operates in DMA mode 0 or operates in DMA mode 1. Note that DMA mode 1 may only be used with bulk endpoints.                     |
|         |          | 0 DMA Mode 0                                                                                                                                                                                                         |
|         |          | 1 DMA Mode 1                                                                                                                                                                                                         |
| 1       | DIR      | DMA Transfer Direction.<br>The USB_DMA <sub>n</sub> _CTL.DIR bit selects the DMA channel transfer direction, which must be selected for use with receive endpoints (DMA write=0) or transmit endpoints (DMA read=1). |
|         |          | 0 DMA Write (for Rx Endpoint)                                                                                                                                                                                        |
|         |          | 1 DMA Read (for Tx Endpoint)                                                                                                                                                                                         |
| 0       | EN       | DMA Enable.<br>The USB_DMA <sub>n</sub> _CTL.EN bit enables the DMA channel starts the DMA transfer.                                                                                                                 |
|         |          | 0 Disable DMA                                                                                                                                                                                                        |
|         |          | 1 Enable DMA (Start Transfer)                                                                                                                                                                                        |

## DMA Channel n Address Register

The USB\_DMA<sub>n</sub>\_ADDR register indicates the location in on-chip memory where DMA data is written or read. The address must be aligned to 32-bit words (The lower two address bits are always zero.) This register increments as the DMA transfer progresses.



USB\_DMAn\_ADDR: DMA Channel n Address Register - R/W

Reset = 0x0000 0000

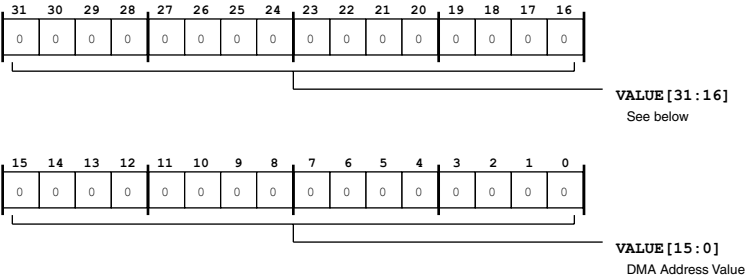


Figure 22-87: USB\_DMAn\_ADDR Register Diagram

Table 22-62: USB\_DMAn\_ADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | DMA Address Value.<br>The USB_DMAn_ADDR.VALUE bits hold the address value for the location in on-chip memory where DMA data is written or read. |

DMA Channel n Count Register

The USB\_DMAn\_CNT register holds the DMA count, indicating the number of bytes to be transferred for a given DMA work block. If this field is set to zero, no data is transferred, and an interrupt is generated.

USB\_DMAn\_CNT: DMA Channel n Count Register - R/W

Reset = 0x0000 0000

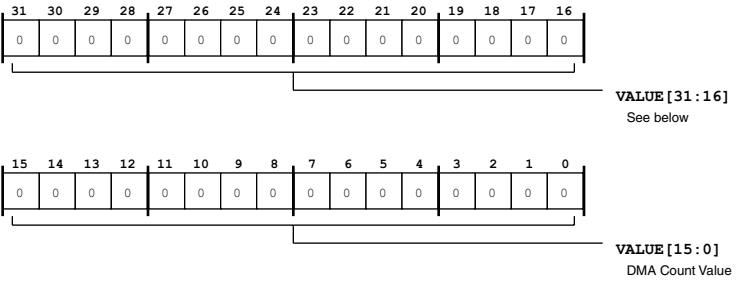


Figure 22-88: USB\_DMAn\_CNT Register Diagram

Table 22-63: USB\_DMA<sub>n</sub>\_CNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | DMA Count Value.<br>The USB_DMA <sub>n</sub> _CNT.VALUE bits indicate the number of bytes to be transferred for a given DMA work block. |

EP<sub>n</sub> Request Packet Count Register

The USB\_RQPKTCNT<sub>n</sub> register specifies (in host mode) the number of packets to request in a block transfer of one or more bulk packets of size USB\_EP<sub>n</sub>\_RXMAXP from a receive endpoint. This register only applies for receive endpoints 1 through 11 in host mode.

USB\_RQPKTCNT<sub>n</sub>: EP<sub>n</sub> Request Packet Count Register - R/W

Reset = 0x0000

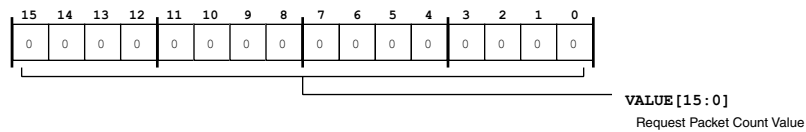


Figure 22-89: USB\_RQPKTCNT<sub>n</sub> Register Diagram

Table 22-64: USB\_RQPKTCNT<sub>n</sub> Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Request Packet Count Value.<br>The USB_RQPKTCNT <sub>n</sub> .VALUE bits specify the number of bulk packets to request in a block transfer from a receive endpoint. This field is used in conjunction with Auto Request feature (USB_EP <sub>n</sub> _RXCSR_H.AUTOREQ). |

Chirp Timeout Register

The USB\_CT\_UCH register selects chirp timeout value. The is value is multiplied by 4 times the XCLK period (or 67ns). The default setting is 1.1ms.

USB\_CT\_UCH: Chirp Timeout Register - R/W

Reset = 0x4074

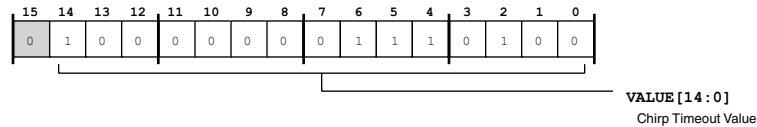


Figure 22-90: USB\_CT\_UCH Register Diagram

Table 22-65: USB\_CT\_UCH Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                           |
|---------|----------|-----------------------------------------------------------------------------------|
| 14:0    | VALUE    | Chirp Timeout Value.<br>The USB_CT_UCH.VALUE bits select the chirp timeout value. |

Host High Speed Return to Normal Register

The USB\_CT\_HHSRTN register selects the delay from end of high speed resume signaling (acting as a host) to return to normal mode operation. This value is multiplied by 4 times the XCLK period (or 16.7ns). The default setting corresponds to a delay of 100us.

USB\_CT\_HHSRTN: Host High Speed Return to Normal Register - R/W

Reset = 0x05e6

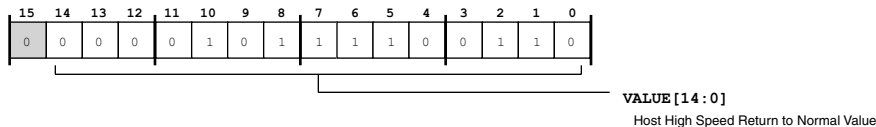


Figure 22-91: USB\_CT\_HHSRTN Register Diagram

Table 22-66: USB\_CT\_HHSRTN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14:0    | VALUE    | Host High Speed Return to Normal Value.<br>The USB_CT_HHSRTN register selects the delay from end of high speed resume signaling (acting as a host) to return to normal mode operation. |

## High Speed Timeout Register

The `USB_CT_HSBT` register selects an amount of time to add to the minimum high speed timeout in units of 64 bit times. The USB 2.0 specification section 7.1.19.2 states that the controller must not timeout less than 736 bit times and must timeout after 816 bit times. The value in `USB_CT_HSBT` is multiplied by 64-bit times and added to the minimum 736 bit times. Settings less than 1 violate the USB 2.0 specification, making the controller non-compliant.

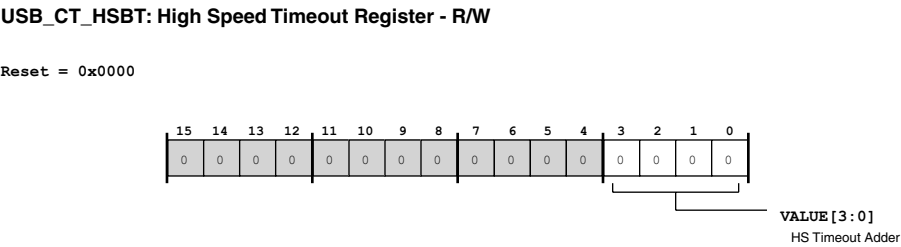


Figure 22-92: USB\_CT\_HSBT Register Diagram

Table 22-67: USB\_CT\_HSBT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | VALUE    | HS Timeout Adder.<br>The <code>USB_CT_HSBT.VALUE</code> bits selects an amount of time to add to the minimum high speed timeout in units of 64 bit times. |
|         |          | 0 HS Timeout = 736 (bit time)                                                                                                                             |
|         |          | 1 HS Timeout = 800 (bit time)                                                                                                                             |
|         |          | 2 HS Timeout = 864 (bit time)                                                                                                                             |

## LPM Attribute Register

The `USB_LPM_ATTR` register defines the link power management (LPM) attributes for LPM transactions and sleep/wake operation. In peripheral mode, the `USB_LPM_ATTR` register contains values received in the most recent, accepted (ACK'd) LPM transaction. In host mode, the `USB_LPM_ATTR` register contains values (loaded by software) that set up the next LPM transaction. The USB controller inserts the LPM values within the next LPM transaction.

**USB\_LPM\_ATTR: LPM Attribute Register - R/W**

Reset = 0x0000

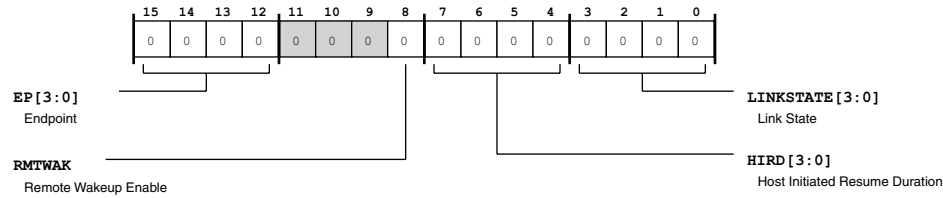


**Figure 22-93:** USB\_LPM\_ATTR Register Diagram

**Table 22-68:** USB\_LPM\_ATTR Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:12   | EP        | Endpoint.<br>The <code>USB_LPM_ATTR.EP</code> bits select the endpoint in the token packet of the LPM transaction.                                                                                                                                                                                                                                                                                            |
| 8       | RMTWAK    | Remote Wakeup Enable.<br>The <code>USB_LPM_ATTR.RMTWAK</code> bit enables remote wakeup. This bit is applied on a temporary basis only and is only applied to the current suspend state. After the current suspend cycle, the remote wakeup capability that was negotiated during enumeration applies.                                                                                                        |
|         | 0         | Disable Remote Wakeup                                                                                                                                                                                                                                                                                                                                                                                         |
|         | 1         | Enable Remote Wakeup                                                                                                                                                                                                                                                                                                                                                                                          |
| 7:4     | HIRD      | Host Initiated Resume Duration.<br>The <code>USB_LPM_ATTR.HIRD</code> bits select the host initiated resume duration. This value is the minimum time that the host drives resume on the bus. The value in this register corresponds to an actual resume time of:<br>$\text{Resume Time} = 50\mu\text{s} + \text{HIRD} \times 75\mu\text{s}$ .<br>This equation produces results in a range of 50us to 1200us. |
| 3:0     | LINKSTATE | Link State.<br>The <code>USB_LPM_ATTR.LINKSTATE</code> bits is value is provided by the host to the peripheral to indicate what state the peripheral must transition to after the receipt and acceptance of a LPM transaction. (Enumerations not shown are reserved.)                                                                                                                                         |
|         | 1         | Sleep State (L1)                                                                                                                                                                                                                                                                                                                                                                                              |

## LPM Control Register

The USB\_LPM\_CTL register controls link power management (LPM) operations, including LPM enable, NAK, resume, and mode transition.

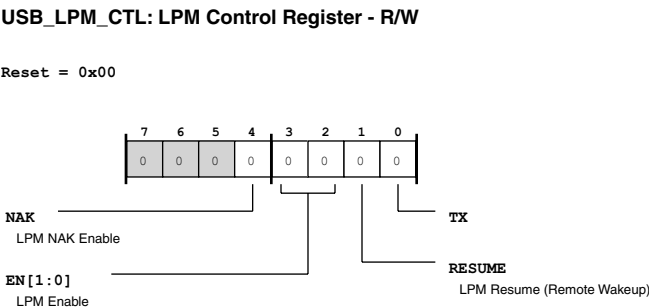


Figure 22-94: USB\_LPM\_CTL Register Diagram

Table 22-69: USB\_LPM\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | NAK      | LPM NAK Enable.<br>The USB_LPM_CTL.NAK bit enables (in peripheral mode) a NAK-all-non-LPM transactions mode for all end points, forcing a NAK response to all transactions other than an LPM transaction. This bit only takes effect after the controller has been LPM suspended. In this case, the USB controller continues to NAK, until this bit has been cleared by software. |
|         |          | 0      Disable LPM NAK                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Enable LPM NAK                                                                                                                                                                                                                                                                                                                                                             |

**Table 22-69:** USB\_LPM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:2     | EN       | LPM Enable.<br>The <code>USB_LPM_CTL.EN</code> bits enable (In peripheral mode) LPM operations. The LPM operation may be enabled at different levels, which determine the response of the USB controller to LPM transactions.                                                                                                                                                                                                                                                                         |
|         |          | 0<br>Disable LPM<br>LPM and extended transactions are not supported. The USB controller does not respond to LPM transactions, and these transaction timeout.                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1<br>Disable LPM<br>LPM and extended transactions are not supported. The USB controller does not respond to LPM transactions, and these transaction timeout.                                                                                                                                                                                                                                                                                                                                          |
|         |          | 2<br>Enable Extended Transactions<br>LPM is not supported, but extended transactions are supported. The USB controller responds to an LPM transaction with a STALL.                                                                                                                                                                                                                                                                                                                                   |
|         |          | 3<br>Enable LPM and Extended Transactions<br>Both LPM and extended transactions are supported. The USB controller responds with a NYET or an ACK as determined by the value of <code>LPMXMT</code> and other conditions.                                                                                                                                                                                                                                                                              |
| 1       | RESUME   | LPM Resume (Remote Wakeup).<br>The <code>USB_LPM_CTL.RESUME</code> bit initiates resume (remote wakeup). This bit's operation differs from the <code>USB_POWER.RESUME</code> bit in that the LPM resume signal timing is controlled by hardware. When set, the USB controller asserts resume signaling for 50us in host mode or asserts resume signaling for the time specified by the <code>USB_LPM_ATTR.HIRD</code> field in device mode. The <code>USB_LPM_CTL.RESUME</code> bit is self clearing. |
|         |          | 0<br>No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1<br>LPM Resume                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Table 22-69:** USB\_LPM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 0       | TX       | <p>.</p> <p>The USB_LPM_CTL.TX bit puts the USB controller in LPM transmit mode, but this mode has differing operations in host mode versus peripheral mode.</p> <p>In peripheral mode, this bit is set by software to instruct the controller to transition to the L1 state upon receipt of the next LPM transaction. This bit is only effective if LPM enable (USB_LPM_CTL.EN) is set to 0x3. The LPM transmit enable bit can be set in the same cycle as LPM enable. If USB_LPM_CTL.TX and USB_LPM_CTL.EN are enabled, the USB controller can respond in the following ways:</p> <ul style="list-style-type: none"><li>• If no data is pending (all transmit FIFOs are empty), the USB controller responds with an ACK, clears the USB_LPM_CTL.TX bit, and generates a software interrupt.</li><li>• If data is pending (data resides in at least one transmit FIFO), the USB controller responds with a NYET, does not clear the USB_LPM_CTL.TX bit, and generates a software interrupt.</li></ul> <p>In host mode, this bit is set by software to transmit an LPM transaction. This bit is self clearing. The USB controller clears this bit immediately on receipt of any token or after three timeouts have occurred.</p> |                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Disable LPM Tx |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Enable LPM Tx  |

## LPM Interrupt Enable Register

The USB\_LPM\_IEN register enables the link power management (LPM) related interrupts. When an interrupt is enabled in this register and the corresponding interrupt is pending in USB\_LPM\_IRQ, the USB controller generates the interrupt. When an interrupt is disabled in this register, the corresponding interrupt may be pending in USB\_LPM\_IRQ, but the USB controller does not generate an interrupt.



**USB\_LPM\_IEN: LPM Interrupt Enable Register - R/W**

Reset = 0x00

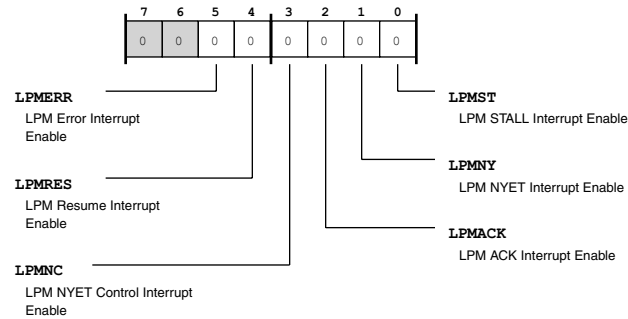


**Figure 22-95:** USB\_LPM\_IEN Register Diagram

**Table 22-70:** USB\_LPM\_IEN Register Fields

| Bit No. | Bit Name | Description/Enumeration            |                   |
|---------|----------|------------------------------------|-------------------|
| 5       | LPMERR   | LPM Error Interrupt Enable.        |                   |
|         |          | 0                                  | Disable Interrupt |
|         |          | 1                                  | Enable Interrupt  |
| 4       | LPMRES   | LPM Resume Interrupt Enable.       |                   |
|         |          | 0                                  | Disable Interrupt |
|         |          | 1                                  | Enable Interrupt  |
| 3       | LPMNC    | LPM NYET Control Interrupt Enable. |                   |
|         |          | 0                                  | Disable Interrupt |
|         |          | 1                                  | Enable Interrupt  |
| 2       | LPMACK   | LPM ACK Interrupt Enable.          |                   |
|         |          | 0                                  | Disable Interrupt |
|         |          | 1                                  | Enable Interrupt  |

Table 22-70: USB\_LPM\_IEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 1       | LPMNY    | LPM NYET Interrupt Enable.  |
|         |          | 0 Disable Interrupt         |
|         |          | 1 Enable Interrupt          |
| 0       | LPMST    | LPM STALL Interrupt Enable. |
|         |          | 0 Disable Interrupt         |
|         |          | 1 Enable Interrupt          |

LPM Interrupt Status Register

The USB\_LPM\_IRQ register indicates link power management (LPM) related interrupt status. The USB controller clears this register when it is read.

USB\_LPM\_IRQ: LPM Interrupt Status Register - R/NW

Reset = 0x00

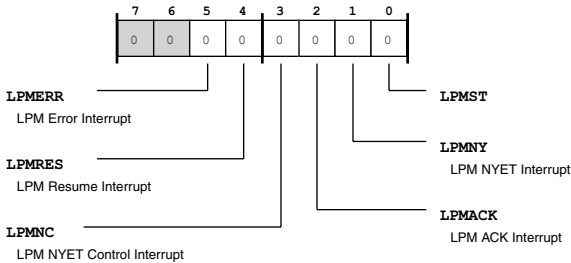


Figure 22-96: USB\_LPM\_IRQ Register Diagram

**Table 22-71:** USB\_LPM\_IRQ Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| 5       | LPMERR   | <p>LPM Error Interrupt.</p> <p>The <code>USB_LPM_IRQ.LPMERR</code> bit indicates an LPM error interrupt condition, but this interrupt has differing conditions for host mode versus peripheral mode.</p> <p>In peripheral mode, this bit is set if an LPM transaction is received that has a LinkState field that is not supported. The USB controller responds to the transaction with a STALL. Note that the USB controller updates the <code>USB_LPM_ATTR</code> register, so software can observe the non compliant LPM packet payload.</p> <p>In host mode, this bit is set if the response to a LPM transaction is received with a bit stuff or PID error. No suspend occurs and the state of the device is now unknown.</p> |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Interrupt Pending    |
| 4       | LPMRES   | <p>LPM Resume Interrupt.</p> <p>The <code>USB_LPM_IRQ.LPMRES</code> bit indicates that the USB controller has been resumed for any reason. This bit is mutually exclusive from the <code>USB_POWER.RESUME</code> bit.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | Interrupt Pending    |

**Table 22-71:** USB\_LPM\_IRQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                      |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| 3       | LPMNC    | <p>LPM NYET Control Interrupt.</p> <p>The <code>USB_LPM_IRQ.LPMNC</code> bit indicates an LPM NYET control interrupt condition, but this interrupt has differing conditions for host mode versus peripheral mode.</p> <p>In peripheral mode, this bit is set when an LPM transaction is received, and the USB controller responds with a NYET due to data pending in the transmit FIFOs. This interrupt may only occur when the <code>USB_LPM_CTL.EN</code> field is set to 11, the <code>USB_LPM_CTL.TX</code> field is set to 1, and there is data pending in the transmit FIFOs.</p> <p>In host mode, this bit is set when an LPM transaction has been transmitted, but has failed to complete. The transaction failure must be because a timeout occurred or be because there were bit errors in the response for three attempts.</p> |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Interrupt Pending    |
| 2       | LPMACK   | <p>LPM ACK Interrupt.</p> <p>The <code>USB_LPM_IRQ.LPMACK</code> bit indicates an LPM ACK interrupt condition, but this interrupt has differing conditions for host mode versus peripheral mode.</p> <p>In peripheral mode, this bit is set when an LPM transaction is received, and the USB controller responds with an ACK. This interrupt may only occur when the <code>USB_LPM_CTL.EN</code> field is set to 11, the <code>USB_LPM_CTL.TX</code> field is set to 1, and there is no data pending in the controller transmit FIFOs.</p> <p>In host mode, this bit is set when an LPM transaction is transmitted, and the device responds with an ACK.</p>                                                                                                                                                                              |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Interrupt Pending    |

**Table 22-71:** USB\_LPM\_IRQ Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| 1       | LPMNY    | LPM NYET Interrupt.<br>The <code>USB_LPM_IRQ.LPMNY</code> bit indicates an LPM NYET interrupt condition, but this interrupt has differing conditions for host mode versus peripheral mode.<br>In peripheral mode, this bit is set when an LPM transaction is received, and the USB controller responds with a NYET. This interrupt may only occur when the <code>USB_LPM_CTL.EN</code> field is set to 11, and the <code>USB_LPM_CTL.TX</code> field is set to 0.<br>In host mode, this bit is set when an LPM transaction is transmitted and the device responds with a NYET. |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Interrupt Pending    |
| 0       | LPMST    | .<br>The <code>USB_LPM_IRQ.LPMST</code> bit indicates an LPM STALL interrupt condition, but this interrupt has differing conditions for host mode versus peripheral mode.<br>This bit is set when an LPM transaction is received, and the USB controller responds with a STALL. This interrupt may only occur when the <code>USB_LPM_CTL.EN</code> field is set to 01.<br>In host mode, this bit is set when an LPM transaction is transmitted, and the device responds with a STALL.                                                                                          |                      |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | No Interrupt Pending |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Interrupt Pending    |

## LPM Function Address Register

The `USB_LPM_FADDR` register selects the link power management (LPM) function address.

**USB\_LPM\_FADDR: LPM Function Address Register - R/W**

Reset = 0x00

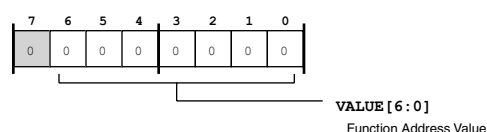


**Figure 22-97:** USB\_LPM\_FADDR Register Diagram

Table 22-72: USB\_LPM\_FADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:0     | VALUE    | Function Address Value.<br>The USB_LPM_FADDR.VALUE bits hold the LPM function address value that the USB controller places in the LPM payload. |

VBUS Control Register

The USB\_VBUS\_CTL controls USB controller VBUS related features.

USB\_VBUS\_CTL: VBUS Control Register - R/W

Reset = 0x00

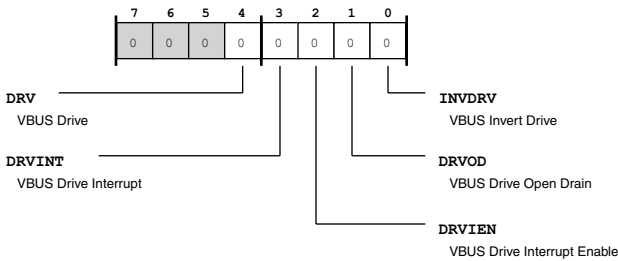


Figure 22-98: USB\_VBUS\_CTL Register Diagram

Table 22-73: USB\_VBUS\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------|
| 4       | DRV      | VBUS Drive.<br>The USB_VBUS_CTL.DRV bit indicates the state of the UTMI+ DrvVBUS signal from the USB controller. |
| 3       | DRVINT   | VBUS Drive Interrupt.<br>The USB_VBUS_CTL.DRVINT bit indicates the state of the DrvVBUSInt interrupt.            |
| 2       | DRVIE    | VBUS Drive Interrupt Enable.<br>The USB_VBUS_CTL.DRVIE bit enables the DrvVBUS interrupt.                        |
| 1       | DRVOD    | VBUS Drive Open Drain.<br>The USB_VBUS_CTL.DRVOD selects whether the DrvVBUS output is open drain.               |

**Table 22-73:** USB\_VBUS\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------|
| 0       | INVDRV   | VBUS Invert Drive.<br>The USB_VBUS_CTL . INVDRV bit selects whether the DrvVBUS output is inverted. |

## Battery Charging Control Register

The USB\_BAT\_CHG controls USB controller battery changing related features.

**USB\_BAT\_CHG: Battery Charging Control Register - R/W**

Reset = 0x00

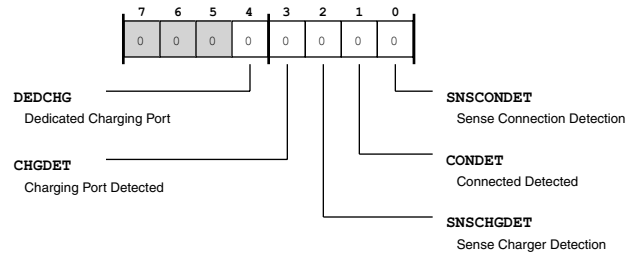


**Figure 22-99:** USB\_BAT\_CHG Register Diagram

**Table 22-74:** USB\_BAT\_CHG Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                  |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DEDCHG    | Dedicated Charging Port.<br>The USB_BAT_CHG . DEDCHG bit is asserted if both D+ and D- are high. This can be used to determine if the attached device is a dedicated charging port. This bit is the decode of LineState[1] and LineState[0]. This bit is only valid when a session is initiated, which enables a pullup on D+ when acting as a B-device. |
| 3       | CHGDET    | Charging Port Detected.<br>The USB_BAT_CHG . CHGDET bit indicates when a charging port is detected. This bit indicates that D+/- is above $V_{DAT\_REF}$ and below $V_{LGC}$ .                                                                                                                                                                           |
| 2       | SNSCHGDET | Sense Charger Detection.<br>The USB_BAT_CHG . SNSCHGDET bit enables charger detection. Setting this bit enables VD_SRC and ID_SINK.                                                                                                                                                                                                                      |

Table 22-74: USB\_BAT\_CHG Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                     |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | CONDET    | Connected Detected.<br>The USB_BAT_CHG.CONDET bit is valid when USB_BAT_CHG.SNSCONDET is enabled. This bit reflects the inverse of D+ (!LineState[0]). If nothing is connected, D+ is pulled high. If a charger or USB port is connected, D+ is pulled low. |
| 0       | SNSCONDET | Sense Connection Detection.<br>The USB_BAT_CHG.SNSCONDET bit enables connection detection. Enabling this bit enables I <sub>DP_SRC</sub> and R <sub>DM_DWN</sub> .                                                                                          |

PHY Control Register

The USB\_PHY\_CTL register provides access to PHY control features.

USB\_PHY\_CTL: PHY Control Register - R/W

Reset = 0x00

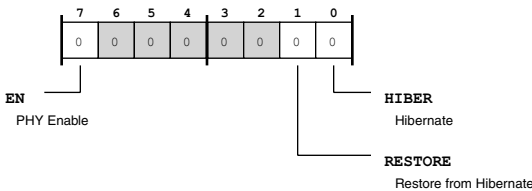


Figure 22-100: USB\_PHY\_CTL Register Diagram

Table 22-75: USB\_PHY\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | EN       | PHY Enable.<br>The USB_PHY_CTL.EN bit enables the USB controller PHY. This bit enables the schmitt-trigger inputs on D+ and D- to detect session request protocol. The bit also enables the bias circuits and VBUS comparators to detect when a host is connected. This bit should be set for all USB controller operations. |



**Table 22-75:** USB\_PHY\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | RESTORE  | Restore from Hibernate.<br>The USB_PHY_CTL.RESTORE bit causes the PHY to come out of hibernate and release its latches.                                 |
| 0       | HIBER    | Hibernate.<br>The USB_PHY_CTL.HIBER bit causes the PHY to prepare for hibernate. Latches hold the pullup/pulldown state when the core power is removed. |

## PLL and Oscillator Control Register

The USB\_PLL\_OSC register provides access to PLL and oscillator related control features.

**USB\_PLL\_OSC: PLL and Oscillator Control Register - R/W**

Reset = 0x0014

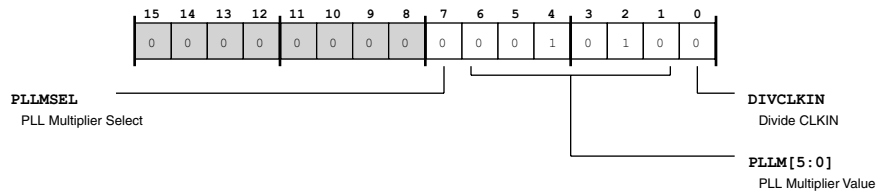


**Figure 22-101:** USB\_PLL\_OSC Register Diagram

**Table 22-76:** USB\_PLL\_OSC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | PLLMSEL  | PLL Multiplier Select.<br>The USB_PLL_OSC.PLLMSEL bit directs the PLL to use the PLL multiplier value is stored the USB_PLL_OSC.PLLM bits.                                                                                          |
| 6:1     | PLLM     | PLL Multiplier Value.<br>The USB_PLL_OSC.PLLM bits hold the PLL multiplier value. This field should be set such that $\text{CLKIN} * \text{USB\_PLL\_OSC.PLLM} = 480\text{MHz}$ . Defaults to $48\text{MHz} * 10 = 480\text{MHz}$ . |

**Table 22-76:** USB\_PLL\_OSC Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------|
| 0       | DIVCLKIN | Divide CLKIN.<br>The <code>USB_PLL_OSC.DIVCLKIN</code> bit enables a divide CLKIN by 2 function for the PLL. |

# 23 Ethernet Media Access Controller (EMAC)

This chapter describes the Ethernet Media Access Controller (EMAC) peripheral. The Ethernet MAC provides a 10/100M bit/s Ethernet interface, compliant to IEEE Std. 802.3-2005, between an RMII (Reduced Media Independent Interface) and the processor.

## Overview

The processor has two identical and independent EMAC peripheral modules, which enables network connectivity to applications via a 10/100M bit/s Ethernet interface. The modules are fully compliant to the following standards:

- Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, Standard 802.3-2005, Institute of Electrical and Electronics Engineers (IEEE).
- Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, Standard 1588-2008, Institute of Electrical and Electronics Engineers (IEEE).
- Reduced Media Independent Interface Specification, Revision 1.2, RMII Consortium.

**NOTE:** Copyright © 2010 Synopsys, Inc.; portions of this chapter are included with permission from Synopsys, Inc.

## Introduction

The EMAC interface consists of necessary hardware for the Media Access Control protocol enabling applications to support TCP/IP based network communication. At the system end, the module supports direct hook up with the System Crossbar bus for Memory/MMR transactions. It supports RMII (Reduced Media Independent Interface) and SMI (Station Management Interface) for interfacing with the external PHY chip.

The MAC also includes a built-in and dedicated DMA controller that performs both data and status transfers between the application and the RMII interface. Internal transmit and receive FIFOs are used to buffer and regulate the frames. A dedicated interrupt line connects the EMAC interrupt sources to the System Event Controller.

The MAC Management Counters (MMC) block is an extended set of registers that collect various statistics compliant with IEEE 802.3 definitions regarding the operation of the interface. They are updated for each new transmitted or received frame when the condition to update the counter is met. The EMAC provides a set of such counters, along with extended usage control.

The EMAC also includes a PTP (Precision Time Protocol) engine that provides hardware assistance for the implementation of the IEEE 1588 Version 1 and Version 2 standards on Blackfin processors, enabling time synchronization between systems.

## EMAC Features

The ethernet MAC's features include the following:

- Supports 10/100 Mbps data transfer rates with external PHY interfaced via RMII.
- Full-Duplex and Half-Duplex support for Ethernet.
- Dedicated DMA controller with independent read write channels.
- Supports dual-buffer (ring) or linked-list (chained) descriptor chaining.
- Direct interface with the System Crossbar bus.
- Provides support for CSMA/CD Protocol for half-duplex operation.
- IEEE 802.3x flow control for Full-Duplex and Half-Duplex.
- Automatic network monitoring statistics with Management Counters.
- Flexible address filtering options for uni-cast/multi-cast/broadcast addresses.
- Support for Promiscuous mode in reception.
- Supports IEEE 802.1Q VLAN tag detection.
- Supports programmable Inter-frame Gap (IFG).
- Checksum Offload Engine for checking IPv4 header checksum and TCP/UDP/ICMP checksum encapsulated in IPv4 or IPv6 datagrams.
- Station Management Interface for PHY device configuration and management.
- Includes FIFOs for buffering: 256 bytes for transmit FIFO and 128 bytes for receive FIFO.
- Automatic CRC and pad generation controllable on a per-frame basis.

## EMAC Functional Description

This section details the functionality of Ethernet MAC peripheral. Sub sections are provided detailing each topic in-depth.

- [\*EMAC Register List\*](#)
- [\*EMAC Definitions\*](#)

- [EMAC Block Diagram and Interfaces](#)
- [EMAC Architectural Concepts](#)

## EMAC Register List

The ethernet MAC (EMAC) module provides a 10/100M bit/s Ethernet interface, compliant to IEEE Std. 802.3-2005, between an RMII (Reduced Media Independent Interface) and the Blackfin processor. A set of registers govern EMAC operations. For more information on EMAC functionality, see the EMAC register descriptions.

**Table 23-1:** EMAC Module Registers

| Name            | Description                  |
|-----------------|------------------------------|
| EMAC_MACCFG     | MAC Configuration Register   |
| EMAC_MACFRMFILT | MAC Rx Frame Filter Register |
| EMAC_HASHTBL_HI | Hash Table High Register     |
| EMAC_HASHTBL_LO | Hash Table Low Register      |
| EMAC_SMI_ADDR   | SMI Address Register         |
| EMAC_SMI_DATA   | SMI Data Register            |
| EMAC_FLOWCTL    | FLOW Control Register        |
| EMAC_VLANTAG    | VLAN Tag Register            |
| EMAC_DBG        | Debug Register               |
| EMAC_ISTAT      | Interrupt Status Register    |
| EMAC_IMSK       | Interrupt Mask Register      |
| EMAC_ADDR0_HI   | MAC Address 0 High Register  |
| EMAC_ADDR0_LO   | MAC Address 0 Low Register   |
| EMAC_MMC_CTL    | MMC Control Register         |
| EMAC_MMC_RXINT  | MMC Rx Interrupt Register    |
| EMAC_MMC_TXINT  | MMC Tx Interrupt Register    |

**Table 23-1:** EMAC Module Registers (Continued)

| Name                | Description                                     |
|---------------------|-------------------------------------------------|
| EMAC_MMC_RXIMSK     | MMC Rx Interrupt Mask Register                  |
| EMAC_MMC_TXIMSK     | MMC TX Interrupt Mask Register                  |
| EMAC_TXOCTCNT_GB    | Tx OCT Count (Good/Bad) Register                |
| EMAC_TXFRMCNT_GB    | Tx Frame Count (Good/Bad) Register              |
| EMAC_TXBCASTFRM_G   | Tx Broadcast Frames (Good) Register             |
| EMAC_TXMCASTFRM_G   | Tx Multicast Frames (Good) Register             |
| EMAC_TX64_GB        | Tx 64-Byte Frames (Good/Bad) Register           |
| EMAC_TX65T0127_GB   | Tx 65- to 127-Byte Frames (Good/Bad) Register   |
| EMAC_TX128T0255_GB  | Tx 128- to 255-Byte Frames (Good/Bad) Register  |
| EMAC_TX256T0511_GB  | Tx 256- to 511-Byte Frames (Good/Bad) Register  |
| EMAC_TX512T01023_GB | Tx 512- to 1023-Byte Frames (Good/Bad) Register |
| EMAC_TX1024TOMAX_GB | Tx 1024- to Max-Byte Frames (Good/Bad) Register |
| EMAC_TXUCASTFRM_GB  | Tx Unicast Frames (Good/Bad) Register           |
| EMAC_TXMCASTFRM_GB  | Tx Multicast Frames (Good/Bad) Register         |
| EMAC_TXBCASTFRM_GB  | Tx Broadcast Frames (Good/Bad) Register         |
| EMAC_TXUNDR_ERR     | Tx Underflow Error Register                     |
| EMAC_TXSNGCOL_G     | Tx Single Collision (Good) Register             |
| EMAC_TXMULTCOL_G    | Tx Multiple Collision (Good) Register           |
| EMAC_TXDEFERRED     | Tx Deferred Register                            |
| EMAC_TXLATECOL      | Tx Late Collision Register                      |
| EMAC_TXEXCESSCOL    | Tx Excess Collision Register                    |
| EMAC_TXCARR_ERR     | Tx Carrier Error Register                       |

**Table 23-1:** EMAC Module Registers (Continued)

| Name                | Description                                     |
|---------------------|-------------------------------------------------|
| EMAC_TXOCTCNT_G     | Tx Octet Count (Good) Register                  |
| EMAC_TXFRMCNT_G     | Tx Frame Count (Good) Register                  |
| EMAC_TXEXCESSDEF    | Tx Excess Deferral Register                     |
| EMAC_TXPAUSEFRM     | Tx Pause Frame Register                         |
| EMAC_TXVLANFRM_G    | Tx VLAN Frames (Good) Register                  |
| EMAC_RXFRMCNT_GB    | Rx Frame Count (Good/Bad) Register              |
| EMAC_RXOCTCNT_GB    | Rx Octet Count (Good/Bad) Register              |
| EMAC_RXOCTCNT_G     | Rx Octet Count (Good) Register                  |
| EMAC_RXBCASTFRM_G   | Rx Broadcast Frames (Good) Register             |
| EMAC_RXMCASTFRM_G   | Rx Multicast Frames (Good) Register             |
| EMAC_RXCRC_ERR      | Rx CRC Error Register                           |
| EMAC_RXALIGN_ERR    | Rx alignment Error Register                     |
| EMAC_RXRUNT_ERR     | Rx Runt Error Register                          |
| EMAC_RXJAB_ERR      | Rx Jab Error Register                           |
| EMAC_RXUSIZE_G      | Rx Undersize (Good) Register                    |
| EMAC_RXOSIZE_G      | Rx Oversize (Good) Register                     |
| EMAC_RX64_GB        | Rx 64-Byte Frames (Good/Bad) Register           |
| EMAC_RX65T0127_GB   | Rx 65- to 127-Byte Frames (Good/Bad) Register   |
| EMAC_RX128T0255_GB  | Rx 128- to 255-Byte Frames (Good/Bad) Register  |
| EMAC_RX256T0511_GB  | Rx 256- to 511-Byte Frames (Good/Bad) Register  |
| EMAC_RX512T01023_GB | Rx 512- to 1023-Byte Frames (Good/Bad) Register |
| EMAC_RX1024TOMAX_GB | Rx 1024- to Max-Byte Frames (Good/Bad) Register |

**Table 23-1:** EMAC Module Registers (Continued)

| Name                    | Description                                    |
|-------------------------|------------------------------------------------|
| EMAC_RXUCASTFRM_G       | Rx Unicast Frames (Good) Register              |
| EMAC_RXLEN_ERR          | Rx Length Error Register                       |
| EMAC_RXOORYPE           | Rx Out Of Range Type Register                  |
| EMAC_RXPAUSEFRM         | Rx Pause Frames Register                       |
| EMAC_RXFIFO_OVF         | Rx FIFO Overflow Register                      |
| EMAC_RXVLANFRM_GB       | Rx VLAN Frames (Good/Bad) Register             |
| EMAC_RXWDOG_ERR         | Rx Watch Dog Error Register                    |
| EMAC_IPC_RXIMSK         | MMC IPC Rx Interrupt Mask Register             |
| EMAC_IPC_RXINT          | MMC IPC Rx Interrupt Register                  |
| EMAC_RXIPV4_GD_FRM      | Rx IPv4 Datagrams (Good) Register              |
| EMAC_RXIPV4_HDR_ERR_FRM | Rx IPv4 Datagrams Header Errors Register       |
| EMAC_RXIPV4_NOPAY_FRM   | Rx IPv4 Datagrams No Payload Frame Register    |
| EMAC_RXIPV4_FRAG_FRM    | Rx IPv4 Datagrams Fragmented Frames Register   |
| EMAC_RXIPV4_UDSBL_FRM   | Rx IPv4 UDP Disabled Frames Register           |
| EMAC_RXIPV6_GD_FRM      | Rx IPv6 Datagrams Good Frames Register         |
| EMAC_RXIPV6_HDR_ERR_FRM | Rx IPv6 Datagrams Header Error Frames Register |
| EMAC_RXIPV6_NOPAY_FRM   | Rx IPv6 Datagrams No Payload Frames Register   |
| EMAC_RXUDP_GD_FRM       | Rx UDP Good Frames Register                    |
| EMAC_RXUDP_ERR_FRM      | Rx UDP Error Frames Register                   |
| EMAC_RXTCP_GD_FRM       | Rx TCP Good Frames Register                    |
| EMAC_RXTCP_ERR_FRM      | Rx TCP Error Frames Register                   |
| EMAC_RXICMP_GD_FRM      | Rx ICMP Good Frames Register                   |



**Table 23-1:** EMAC Module Registers (Continued)

| Name                    | Description                                  |
|-------------------------|----------------------------------------------|
| EMAC_RXICMP_ERR_FRM     | Rx ICMP Error Frames Register                |
| EMAC_RXIPV4_GD_OCT      | Rx IPv4 Datagrams Good Octets Register       |
| EMAC_RXIPV4_HDR_ERR_OCT | Rx IPv4 Datagrams Header Errors Register     |
| EMAC_RXIPV4_NOPAY_OCT   | Rx IPv4 Datagrams No Payload Octets Register |
| EMAC_RXIPV4_FRAG_OCT    | Rx IPv4 Datagrams Fragmented Octets Register |
| EMAC_RXIPV4_UDSBL_OCT   | Rx IPv4 UDP Disabled Octets Register         |
| EMAC_RXIPV6_GD_OCT      | Rx IPv6 Good Octets Register                 |
| EMAC_RXIPV6_HDR_ERR_OCT | Rx IPv6 Header Errors Register               |
| EMAC_RXIPV6_NOPAY_OCT   | Rx IPv6 No Payload Octets Register           |
| EMAC_RXUDP_GD_OCT       | Rx UDP Good Octets Register                  |
| EMAC_RXUDP_ERR_OCT      | Rx UDP Error Octets Register                 |
| EMAC_RXTCP_GD_OCT       | Rx TCP Good Octets Register                  |
| EMAC_RXTCP_ERR_OCT      | Rx TCP Error Octets Register                 |
| EMAC_RXICMP_GD_OCT      | Rx ICMP Good Octets Register                 |
| EMAC_RXICMP_ERR_OCT     | Rx ICMP Error Octets Register                |
| EMAC_TM_CTL             | Time Stamp Control Register                  |
| EMAC_TM_SUBSEC          | Time Stamp Sub Second Increment Register     |
| EMAC_TM_SEC             | Time Stamp Low Seconds Register              |
| EMAC_TM_NSEC            | Time Stamp Nano Seconds Register             |
| EMAC_TM_SECUPDT         | Time Stamp Seconds Update Register           |
| EMAC_TM_NSECUPDT        | Time Stamp Nano Seconds Update Register      |
| EMAC_TM_ADDEND          | Time Stamp Addend Register                   |

**Table 23-1:** EMAC Module Registers (Continued)

| Name                 | Description                                   |
|----------------------|-----------------------------------------------|
| EMAC_TM_TGTM         | Time Stamp Target Time Seconds Register       |
| EMAC_TM_NTGTM        | Time Stamp Target Time Nano Seconds Register  |
| EMAC_TM_HISEC        | Time Stamp High Second Register               |
| EMAC_TM_STMPSTAT     | Time Stamp Status Register                    |
| EMAC_TM_PPSCTL       | PPS Control Register                          |
| EMAC_TM_AUXSTMP_NSEC | Time Stamp Auxiliary TS Nano Seconds Register |
| EMAC_TM_AUXSTMP_SEC  | Time Stamp Auxiliary TM Seconds Register      |
| EMAC_TM_PPSINTVL     | Time Stamp PPS Interval Register              |
| EMAC_TM_PPSWIDTH     | PPS Width Register                            |
| EMAC_DMA_BUSMODE     | DMA Bus Mode Register                         |
| EMAC_DMA_TXPOLL      | DMA Tx Poll Demand Register                   |
| EMAC_DMA_RXPOLL      | DMA Rx Poll Demand register                   |
| EMAC_DMA_RXDSC_ADDR  | DMA Rx Descriptor List Address Register       |
| EMAC_DMA_TXDSC_ADDR  | DMA Tx Descriptor List Address Register       |
| EMAC_DMA_STAT        | DMA Status Register                           |
| EMAC_DMA_OPMODE      | DMA Operation Mode Register                   |
| EMAC_DMA_IEN         | DMA Interrupt Enable Register                 |
| EMAC_DMA_MISS_FRM    | DMA Missed Frame Register                     |
| EMAC_DMA_RXIWDOG     | DMA Rx Interrupt Watch Dog Register           |
| EMAC_DMA_BMMODE      | DMA SCB Bus Mode Register                     |
| EMAC_DMA_BMSTAT      | DMA SCB Status Register                       |
| EMAC_DMA_TXDSC_CUR   | DMA Tx Descriptor Current Register            |

**Table 23-1:** EMAC Module Registers (Continued)

| Name               | Description                        |
|--------------------|------------------------------------|
| EMAC_DMA_RXDSC_CUR | DMA Rx Descriptor Current Register |
| EMAC_DMA_TXBUF_CUR | DMA Tx Buffer Current Register     |
| EMAC_DMA_RXBUF_CUR | DMA Rx Buffer Current Register     |

## ADSP-BF60x EMAC Interrupt List

**Table 23-2:** ADSP-BF60x EMAC Interrupt List

| Description  | ID | DMA Channel | Sensitivity |
|--------------|----|-------------|-------------|
| EMAC0 Status | 68 |             | LEVEL       |
| EMAC1 Status | 70 |             | LEVEL       |

## EMAC Definitions

The following definitions aid with using the EMAC.

### ***EMAC SCB***

System Crossbar interface of EMAC

### ***EMAC DMA***

DMA Controller of EMAC

### ***EMAC MFL***

MAC FIFO Layer inside EMAC

### ***EMAC CORE***

CORE Layer inside EMAC which does the actual Ethernet operations, including interface with PHY via RMII.

### ***MMC***

MAC Management Counter

### ***SMI***

Station Management Interface that controls PHY via MDIO/MDC signals.

### ***MII***

Reduced Media Independent Interface

### ***MAC***

Media Access Control

### ***PTP***

Precision Time Protocol

## EMAC Block Diagram and Interfaces

The below figure illustrates the overall functional architecture of the Ethernet MAC peripheral. The EMAC module basically comprises of four major layers, EMAC SCB, EMAC DMA, EMAC MFL and EMAC CORE. Each of these layers (sub-blocks) are explained in depth in their respective sections in this chapter.

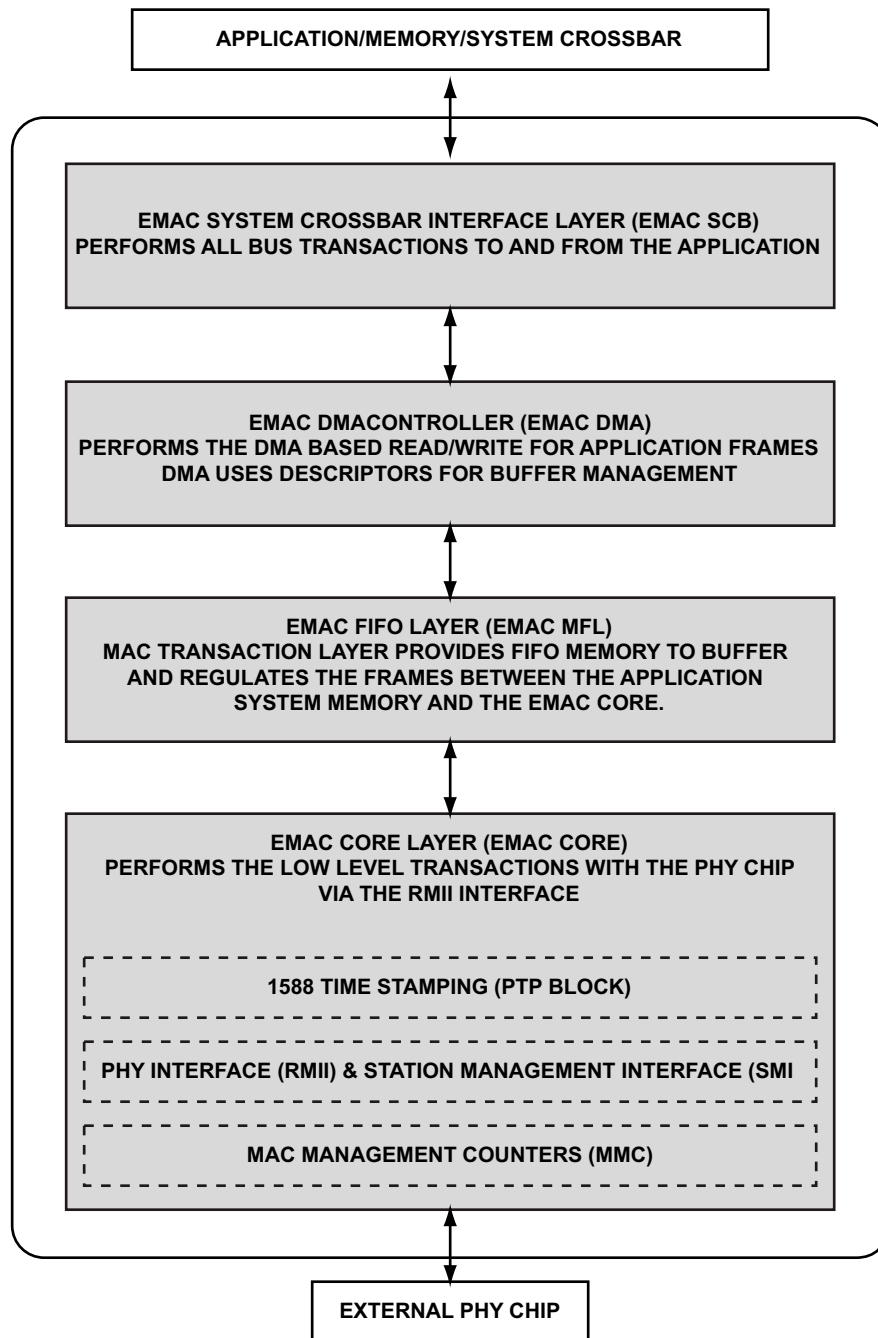


**Figure 23-1:** EMAC Simplified Block Diagram

A more comprehensive block diagram is shown next, which includes most of the important blocks inside the EMAC. The EMAC is connected to processor memory and system crossbar via the System Crossbar Bus Interface (SCB) and System Peripheral Bus Interface (SPB), which are part of the SCB Layer. The SPB interface is connected to all modules that require MMR programming. The DMA controller performs application data transfer frame by frame, through well defined descriptor structures. A FIFO layer acts as a buffer between the DMA controller and the EMAC core. The EMAC core is the most important block as

it contains sub-blocks to support IEEE802.3 based communication with external network interfaces of 10/100 Mbps speeds. It includes the PTP sub-block to assist applications requiring time synchronization and a MMC sub-block to generate packet transfer statistics. The MAC is connected to the external PHY via the Reduced Media Independent Interface (RMII) and the Station Management Interface (Serial Management IO).

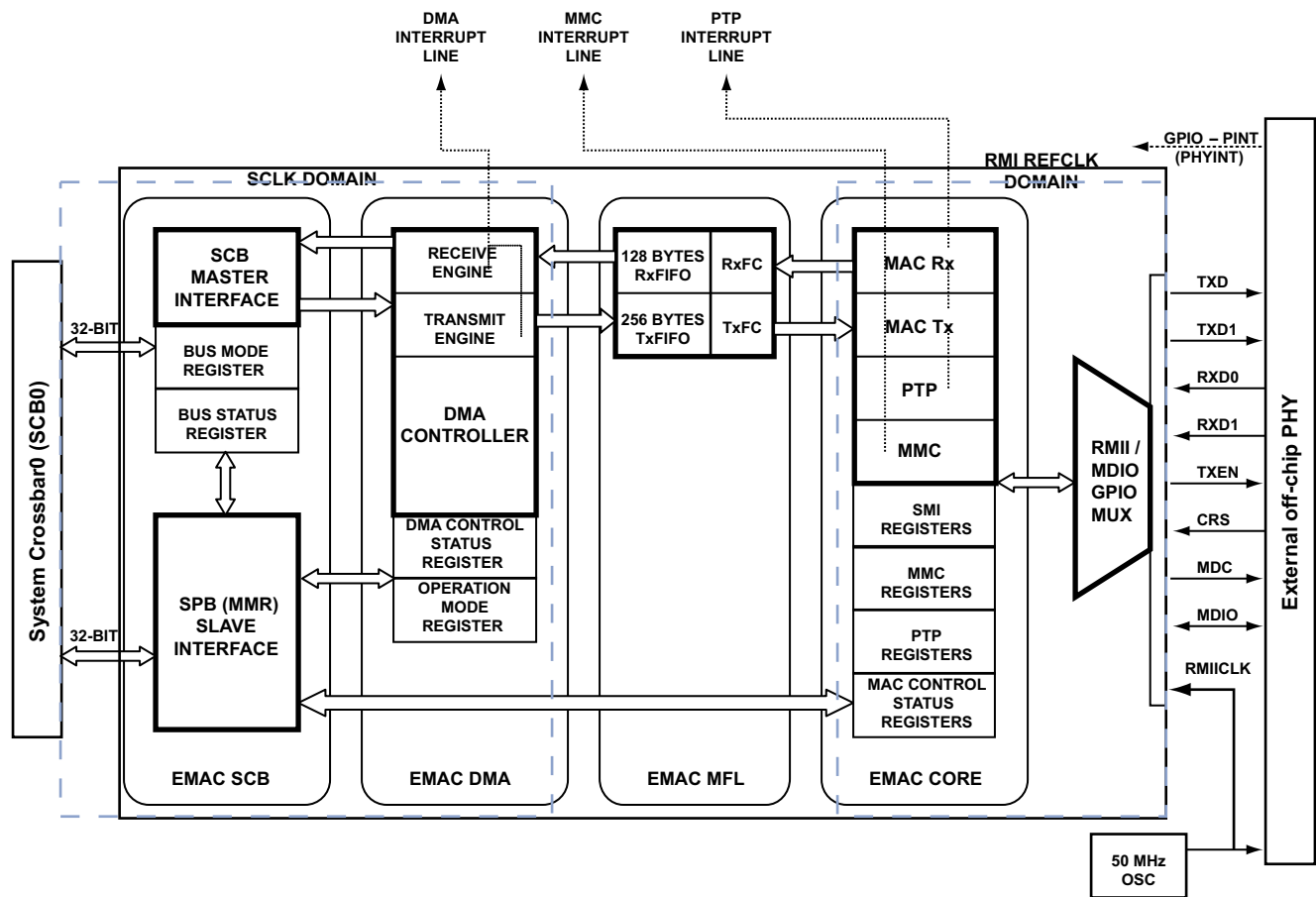


Figure 23-2: EMAC Complete Block Diagram

EMAC CORE Sub-blocks

The core transmit engine sub-blocks and their function are summarized in the below table. Please refer to the EMAC core section for further explanation of each of these sub-blocks.

Table 23-3: Core Transmit Engine sub-blocks

| CORE Transmit Engine Sub Block | Function               |
|--------------------------------|------------------------|
| Transmit Bus Interface         | Interface to the FIFO. |

**Table 23-3:** Core Transmit Engine sub-blocks (Continued)

| CORE Transmit Engine Sub Block   | Function                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transmit Frame Controller        | <ul style="list-style-type: none"> <li>–Appends Zero-PAD data if required, for short frames.</li> <li>–Appends CRC for Frame Check-Sum from the CRC Generator.</li> </ul>                                                                                                                                                                  |
| Transmit Protocol Engine         | <ul style="list-style-type: none"> <li>–Generates preamble and SFD, as per 802.3 protocol.</li> <li>–Generates jam pattern in Half-Duplex mode, for collisions.</li> <li>–Jabber timeout, for excessively large frames.</li> <li>–Flow control for Half-Duplex mode (back pressure).</li> <li>–Generates transmit frame status.</li> </ul> |
| Transmit Scheduler               | <ul style="list-style-type: none"> <li>–Maintains the inter-frame gap between two transmitted frames.</li> <li>–Follows the Truncated Binary Exponential Back-off algorithm for Half-Duplex mode.</li> </ul>                                                                                                                               |
| Transmit CRC Generator           | Generate CRC for the Frame Check-Sum field of the Ethernet frame.                                                                                                                                                                                                                                                                          |
| Transmit Flow Control            | Receives the Pause frame, appends the calculated CRC, and sends the frame to the Protocol Engine module.                                                                                                                                                                                                                                   |
| Transmit Checksum Offload Engine | Supports checksum calculation and insertion in the transmit path, for IPV4/TCP/UDP/ICMP packets.                                                                                                                                                                                                                                           |

The core receive engine sub-blocks and their function are summarized in the following table. Please refer to the EMAC core section for more information on each of these sub-blocks.

**Table 23-4:** Core Receive Engine sub-blocks

| CORE Receive Engine sub block | Functionality overview                                                                                                                                                                                                                                                                                          |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Receive Protocol Engine       | <ul style="list-style-type: none"> <li>–Strips the incoming preamble and SFD.</li> <li>–Checks for correct Length/Type field.</li> <li>–Performs internal loopback if required.</li> <li>–Generates receive status.</li> <li>–Supports watchdog of received frames.</li> <li>–Supports Jumbo Frames.</li> </ul> |
| Receive CRC Module            | Checks for CRC error, by comparing with FCS.                                                                                                                                                                                                                                                                    |

**Table 23-4:** Core Receive Engine sub-blocks (Continued)

| CORE Receive Engine sub block      | Functionality overview                                                                                                                                                                                                                                                                                                                 |
|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Receive Frame Controller Module    | <ul style="list-style-type: none"> <li>–Packs incoming 8-bit input stream to 32-bit data internally.</li> <li>–Performs Frame filtering, for uni-cast/multi-cast/broadcast frames.</li> <li>–Attaches the calculated IP Checksum input from Checksum Offload Engine.</li> <li>–Updates the Receive Status to Bus Interface.</li> </ul> |
| Receive Flow Control Module        | <ul style="list-style-type: none"> <li>–Detects the receiving Pause frame and pauses the frame transmission for the delay specified within the received Pause frame.</li> <li>–Works in Full Duplex mode.</li> </ul>                                                                                                                   |
| Receive IP Checksum Offload Engine | <ul style="list-style-type: none"> <li>–Calculates IPv4 header checksums and verify against the received IPv4 header checksums.</li> <li>–Identifies a TCP, UDP or ICMP payload in the received IP datagrams.</li> </ul>                                                                                                               |
| Receive Bus Interface Unit Module  | Interface to the FIFO.                                                                                                                                                                                                                                                                                                                 |
| Address Filtering Module           | <p>Performs Destination Address Filtering based on Unicast/ Multi-cast/Broadcast frames.</p> <ul style="list-style-type: none"> <li>–Provides CRC hash filtering.</li> </ul>                                                                                                                                                           |

## PHY Interface

The EMAC can be interfaced to the PHY via the RMII interface standard. The tables below indicate the RMII pins available in the EMAC in terms of their generic names. Please refer to the data sheet for exact pin names.

**Table 23-5:** RMII pins

| Sl. No. | Generic Signal Name (IEEE Standards) | RMII Pin functionality.                   |
|---------|--------------------------------------|-------------------------------------------|
| 1.      | TXD0                                 | RMII transmit data pin D0 (di-bit lower)  |
| 2.      | TXD1                                 | RMII transmit data pin D1 (di-bit higher) |
| 3.      | RXD0                                 | RMII receive data pin D0 (di-bit lower)   |



**Table 23-5:** RMII pins (Continued)

| Sl. No. | Generic Signal Name (IEEE Standards) | RMII Pin functionality.                                        |
|---------|--------------------------------------|----------------------------------------------------------------|
| 4.      | RXD1                                 | RMII receive data pin D1 (di-bit higher)                       |
| 5.      | RMII CLK                             | RMII common clock (for Tx and Rx), also called reference clock |
| 6.      | TXEN                                 | RMII transmit enable pin (Tx valid)                            |
| 7.      | CRS                                  | RMII Carrier Sense / receive data valid                        |
| 8.      | MDC                                  | Serial management clock driven by EMAC                         |
| 9.      | MDIO                                 | Serial management bi-directional data                          |

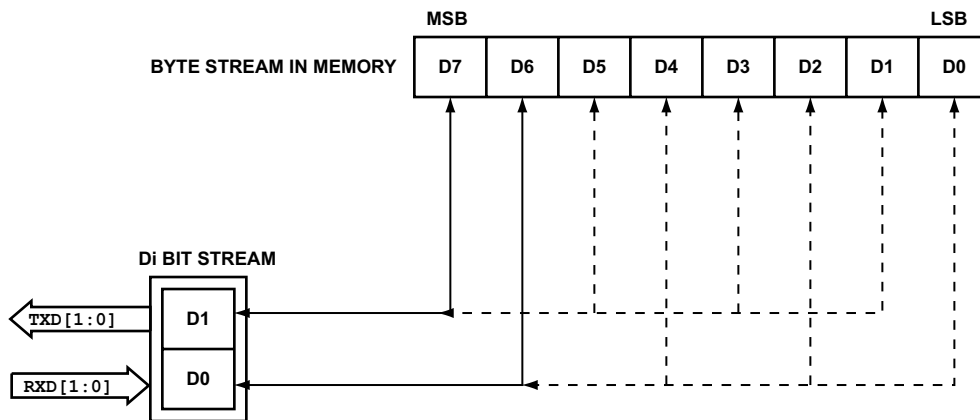


**Figure 23-3:** RMII di-bit data transfer

## EMAC Clock Sources

The Ethernet MAC is clocked internally from SCLK. Check the processor data sheet for the valid frequency range of the appropriate SCLK signal for Ethernet operation.

A 50 MHz clock should be sourced externally to operate the EMAC in RMII mode. This clock is same for both transmission and reception. The MDC Station Management Clock is derived from the SCLK and driven from the MAC to the PHY, when accessing any PHY registers.

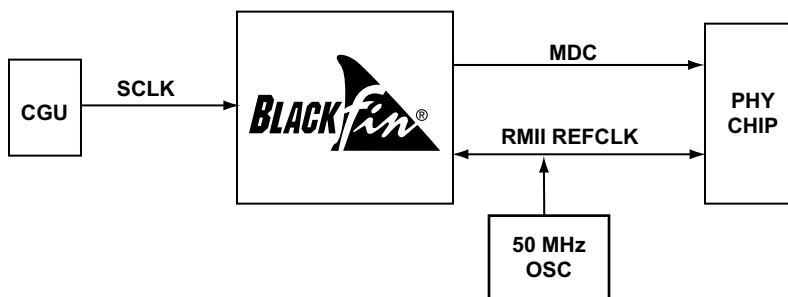


**Figure 23-4:** EMAC Clock Sources

## EMAC Architectural Concepts

This section explains different architectural concepts relevant to EMAC peripheral, such as EMAC SCB, EMAC DMA EMAC MFL, EMAC CORE and others.

### EMAC System Crossbar Interface (EMAC SCB)

The EMAC SCB bus interface provides the bus connectivity to support highly effective data traffic throughput. The system bus use is maximized by allowing simultaneous read and write transfers initiated from different DMA channels. The EMAC controller is directly hooked up to the SCB0 crossbar of BF60x. the following interfaces are available with the design:

- A 32 bit SCB master interface for reading/writing from/to application memory.
- A 32 bit SPB slave interface for register programming.

Please refer to the System Crossbars (SCB) chapter for more information on how the crossbar operates. Only the EMAC specific information is detailed in this chapter.

**Table 23-6:** EMAC-SCB Interface Data Transfer Specifications with Crossbar

| Specification Term | Comments                             |
|--------------------|--------------------------------------|
| 1 beat in SCB      | SINGLE burst                         |
| BLEN4 bursts       | 4 beats in SCB                       |
| BLEN8 bursts       | 8 beats in SCB                       |
| BLEN16 bursts      | 16 beats in SCB                      |
| Bus size           | 32-bit fixed bus size; equals 1 beat |
| INCR bursts        | Incrementing Bursts                  |

**Table 23-6:** EMAC-SCB Interface Data Transfer Specifications with Crossbar (Continued)

| Specification Term  | Comments                          |
|---------------------|-----------------------------------|
| INCR ALIGNED bursts | Incrementing aligned bursts       |
| UNDEF bursts        | Undefined burst length            |
| PBL                 | Programmable Burst Length for DMA |

DMA Write Channel and Read Channel data paths with the system crossbar is shown below.

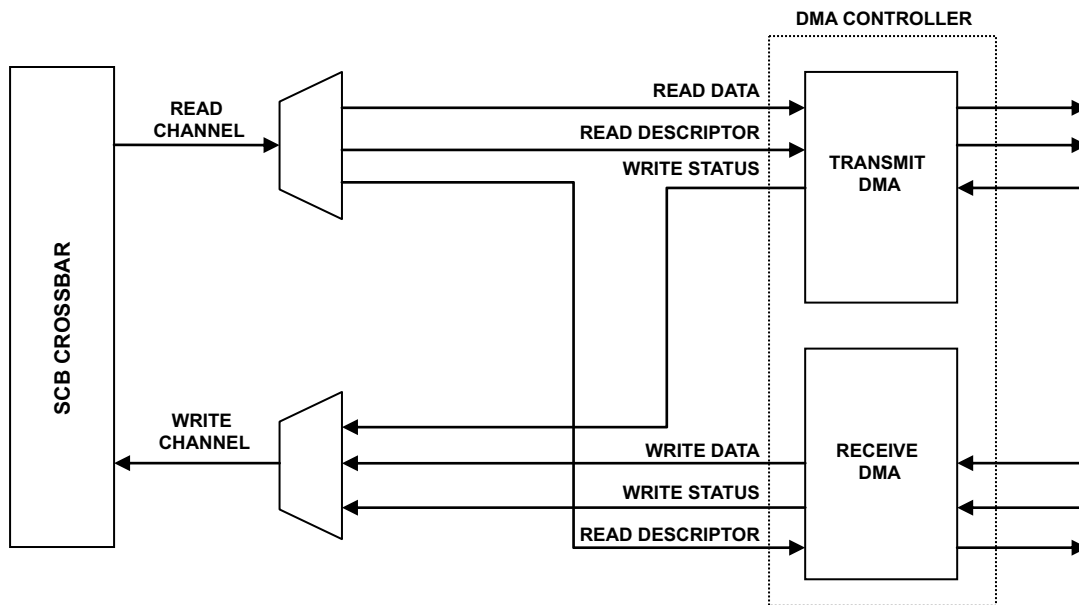


**Figure 23-5:** EMAC DMA Read/Write channels with System Crossbar

**NOTE:** Transmit descriptor read and receive descriptor write-back (status update) operation can occur simultaneously. However transmit descriptor read and transmit descriptor write-back operation cannot occur simultaneously as transmit DMA (or receive DMA) does not initiate the next transfer unless the previous one is complete.

### Priority of SCB Requests

The Descriptor transfers have higher priority than the data transfers. So, if there are two bus requests, for example, receive descriptor read and transmit data read, then receive descriptor read is given higher priority so, that the next receive data write (subsequent to receive descriptor read) need not wait for transmit data read transfer to complete. If there are descriptor read requests from both DMA channels, then they are serviced based on a first-come first-serve basis. Receive DMA has higher priority if descriptor read requests are generated from both the DMAs in the same clock. Similarly, in the write channel, descriptor writes from any DMA have higher priority than the data-write transfers for the Receive DMA.

## SCB Interface Programming Options

The SCB bus interface supports the following options for the EMAC module. The `DMA_BMMODE` register together with the `DMA_BUSMODE` register controls the programmable features of the SCB master interface.

- **Outstanding Transactions:** The EMAC-SCB supports up to four Read/Write outstanding requests on the SCB bus. This can also be controlled through software by programming the `WROSRLMT`/`RDOSRLMT` bits in the DMA Bus Mode Register. Maximum outstanding requests = `WROSRLMT` + 1 (or) `RDOSRLMT` + 1.
- **Allowed burst sizes:** 4 (`BLLEN4`), 8 (`BLLEN8`), 16 (`BLLEN16`) and the `SINGLE` burst. Only those burst sizes configured by user (via `DMA_BMMODE` register) are used for data transfer through the SCB bus. However, `SINGLE` burst is available by default, when the `UNDEF` bit in the `DMA_BMMODE` register is cleared. Data transfers are restricted to the maximum burst size from this list of programmed burst sizes.
- **Burst Splitting and Burst Selection:** The EMAC-SCB splits the DMA requests into multiple bursts on the SCB System Bus. The splitting is based on DMA count and software controllable burst enable bits `UNDEF`, `BLLEN16`, `BLLEN8`, and `BLLEN4` as well as burst types (`INCR` and `INCR_ALIGNED`) which are also controllable through the software. `SINGLE` burst is enabled when the `UNDEF` bit is not set. Burst length select priority is in the sequence: `UNDEF`, `BLLEN16`, `BLLEN8`, and `BLLEN4`.
- **INCR Burst Type:**
  - If the `UNDEF` bit in `DMA_BMMODE` register is set, then EMAC-SCB always chooses the maximum allowed burst length based on the `BLLEN16`, `BLLEN8`, `BLLEN4` bits. In cases where the DMA requests are not multiples of the maximum allowed burst length, the SCB may also choose a burst-length of any value less than the maximum enabled burst-length (all lesser burst-length enables are redundant). For example, when `BLLEN16`, `BLLEN8`, `BLLEN4` are enabled and the DMA requests a burst transfer size of 42 beats, then the SCB splits it into three bursts of 16, 16 and 10 beats respectively.
  - If `UNDEF` is not enabled, then the burst length is based on the priority of the enabled bits in the following order (`BLLEN16`, `BLLEN8`, and `BLLEN4`). When the DMA requests a burst transfer, the SCB interface splits the requested bursts into multiple transfers using only the enabled burst lengths. This splitting can occur when the requested burst is not a multiple of the maximum enabled burst. If it cannot choose any of the enabled burst lengths then it selects the burst length as 1. For example, when `BLLEN16`, `BLLEN8`, `BLLEN4` are enabled and the DMA requests a burst transfer of 42 beats, then the SCB interface splits it into multiple bursts of size 16, 16, 8, 1 and 1 beats respectively (the sequence is in decreasing burst sizes).
- **INCR\_ALIGNED Burst Type:** When the address-aligned burst-type is enabled (`AAL` bit in `DMA_BMMODE` register), then in addition to the burst splitting conditions explained in the `INCR` Burst type, the SCB interface splits the DMA requested bursts such that each burst-size is aligned to the least significant bits of the start address. The SCB interface initially generates smaller bursts so that the remaining transfers can be transferred with the maximum possible (enabled) fixed burst lengths. For example, in the same setting as explained earlier for `UNDEF` set (`BLLEN16`, `BLLEN8` and `BLLEN4` are enabled), DMA requests a burst size of 42 beats at the start address of `0x000003A4`. The SCB starts the first transfer with size 3 such that the address of the next burst is aligned (`0x000003B0`) for a burst of 16. Therefore, the sequence of bursts would be 3, 16, 16, and 7, respectively. When `UNDEF` is not set, then in the same scenario (start address

of 0x000003A4 with 42 beats), the sequence of burst transfers would be 1, 1, 1, 16, 16, 4, and 3 respectively. The sequence of smaller bursts at the beginning is to align the address to the next higher enabled burst-lengths programmed in the register.

- Burst operations for DMA transactions: PBL (Programmable Burst Length) field of the DMA\_BUSMODE [13:8] register indicate the maximum number of beats to be transferred in one DMA transaction. This is also the maximum value that is used in a single block read/write.

**Table 23-7:** DMA PBL Max Limits

| Burst Limit max term     | Definition                    |
|--------------------------|-------------------------------|
| PBL-max limit            | (FIFO size / 2) / 4 words.    |
| PBL-max limit (transmit) | 256 bytes / 2 / 4 = 32 words. |
| PBL-max limit (receive)  | 128 bytes / 2 / 4 = 16 words. |

- For example, if PBL = 32 and if BLEN16 is enabled, the DMA automatically splits 32 bursts in to 2 x 16 bursts. If PBL = 8, and if BLEN16 and BLEN8 are enabled, the maximum burst is limited to BLEN8. If the PBL8 bit in the DMA\_BUSMODE register is set, the programmed PBL value is multiplied by 8 times internally. However, the result cannot be more than the maximum limits specified above.
- The Receive DMA burst length configuration can be made independent of transmit DMA configuration, by setting the bit in the register. If this bit is set, bits defines the burst length of receive DMA. If bit is not set, the bits are used for both transmit and receive. Programs must ensure that the PBL maximum limit is not violated.
- The Receive and Transmit descriptors are always accessed in the maximum possible burst-size (limited by PBL maximum for transmit and receive) for the 16-bytes to be read.

## DMA Bursts with SCB Interface

The Transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in the Transmit FIFO or the number of bytes until the end of frame (when it is less than the configured burst-length). The DMA indicates the start address and the number of transfers required to the SCB master interface. When the SCB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4/8/16 and 1 beat transaction.

The Receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in the MTL Receive FIFO or when the end of frame (when it is less than the configured burst-length) is detected in the Receive FIFO. The DMA indicates the start address and the number of transfers required to the SCB master interface. When the SCB interface is configured for fixed-length burst, then it transfers data using the best combination of INCR 4/8/16 or 1 beat transaction. If the end-of frame is reached before the fixed-burst ends on the SCB interface, then dummy transfers are performed in-order to complete the fixed-burst. Otherwise (if DMA\_BUSMODE.FB is reset), it transfers data using INCR (undefined length) transactions.

When the SCB interface is configured for address-aligned beats (AAL), both DMA engines ensure that the first burst transfer the SCB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL.

## SCB Bus Transaction Status

The `BUSRD` and `BUSWR` bits of the `DMA_BMSTAT` register give the status of SCB bus activities. These bits indicate whether the channel is active or not.

## Fatal Bus Error

The EMAC SCB asserts the error interrupt when the corresponding Fatal Bus Error interrupt is enabled in the DMA interrupt enable register. The application has to reset the core to restart the DMA.

## EMAC DMA Controller (EMAC DMA)

The EMAC has an in-built DMA controller that performs reads and writes of application data and descriptors via the SCB master interface.

The DMA has independent transmit and receive engines, and a CSR (control and status register) space. The transmit engine transfers data from system memory to a FIFO, while the receive engine transfers data from the FIFO to the system memory. The controller uses a descriptor chain based transfer mechanism to efficiently move data from source to destination with minimal processor core intervention. The DMA is specially designed for packet-oriented data transfers such as Ethernet frames. The controller can be programmed to interrupt the application for situations such as frame transmit and receive transfer completion, and other normal or abnormal conditions.

The DMA and the application device driver communicate through two internal data structures:

1. DMA Control and Status registers (CSR).
2. Descriptor lists and data buffers.

Descriptors that reside in the application memory act as pointers to receive and transmit buffers. There are two descriptor lists, one for receive, and one for transmit. The base address of each list is written into the receive descriptor list address register and transmit descriptor list address register, respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both receive and transmit descriptors. The descriptor lists reside in the application memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers may contain only data and buffer status is maintained in the descriptor itself. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA will skip to the next frame buffer when the end-of-frame is detected. Data chaining can be enabled or disabled.

As mentioned above, the controller supports two types of descriptors: ring mode and chain mode. Their functional structure as provided to the user is shown in below figures.

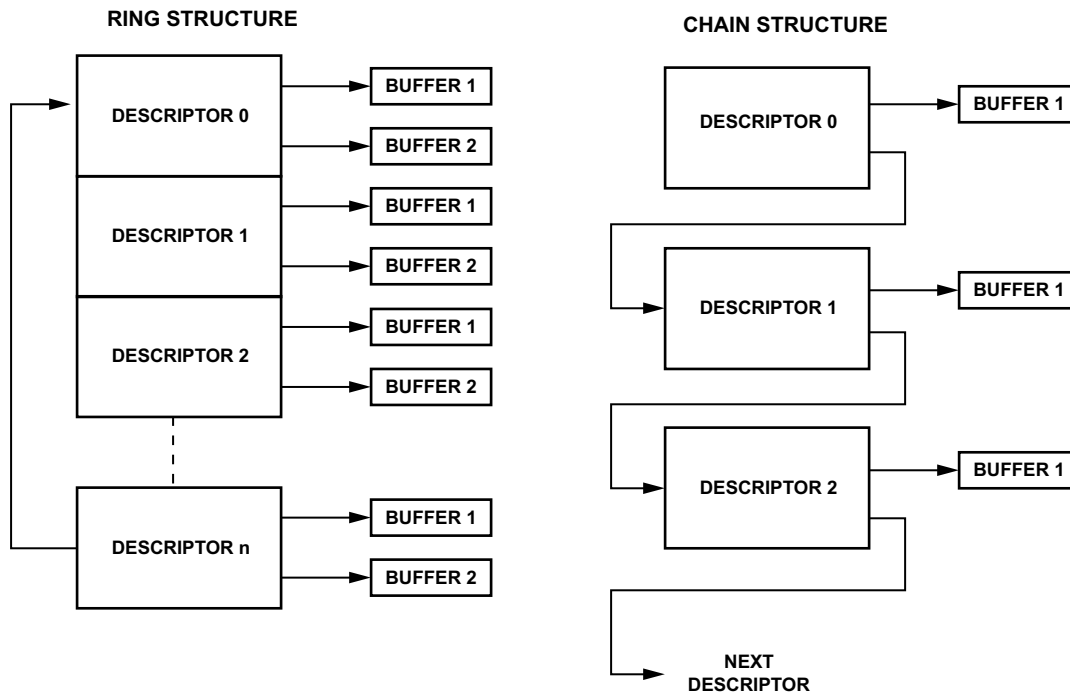


**Figure 23-6:** EMAC DMA Descriptor Models

**NOTE:** It is possible to define a skip length (in terms of  $N \times 32$ -bit words) between two subsequent descriptors, when using ring mode. The DSL field in the DMA bus mode register must be programmed to enable this. With this option available, programs are not always restricted to a contiguous memory location in ring mode.

## DMA Related Registers

A summary of DMA registers relative to their functionality is given below. Please refer to the Register Descriptions sections for complete bit descriptions of each of these registers.

**Table 23-8:** Summary of DMA related Registers.

| Register Name     | Related Functionality                                                                                           |
|-------------------|-----------------------------------------------------------------------------------------------------------------|
| Bus Mode Register | The Bus Mode Register establishes the bus operating modes for the DMA with respect to the SCB master interface. |

**Table 23-8:** Summary of DMA related Registers. (Continued)

| Register Name                                     | Related Functionality                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Transmit Poll Demand Register                     | The Transmit Poll Demand Register enables the transmit DMA to check whether or not the current descriptor is owned by DMA. The Transmit Poll Demand command is given to wake up the TxDMA if it is in Suspend mode. The TxDMA can go into suspend mode because of an Underflow error in a transmitted frame or because of the unavailability of descriptors owned by transmit DMA. You can give this command anytime and the TxDMA resets this command once it starts re-fetching the current descriptor from host memory. |
| Receive Poll Demand Register                      | The Receive Poll Demand Register enables the receive DMA to check for new descriptors. This command is given to wake up the RxDMA from SUSPEND state. The RxDMA can go into SUSPEND state only because of the unavailability of descriptors owned by it.                                                                                                                                                                                                                                                                   |
| Receive Descriptor List Address Register          | The Receive Descriptor List Address Register points to the start of the Receive Descriptor List. The descriptor lists reside in the application memory space and must be Word -aligned (32- bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low.                                                                                                                                                                                                            |
| Transmit Descriptor List Address Register         | The Transmit Descriptor List Address Register points to the start of the Transmit Descriptor List. The descriptor lists reside in the application memory space and must be Word (for 32-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.                                                                                                                                                                                                                 |
| DMA Status Register                               | The Status Register contains all the status bits that the DMA reports to the application. The Software driver reads this register during an interrupt service routine or polling. Most of the fields in this register cause the host to be interrupted.                                                                                                                                                                                                                                                                    |
| Operation Mode Register                           | The Operation Mode Register establishes the Transmit and Receive operating modes and commands. Operation Mode Register should be the last Control Register to be written as part of DMA initialization.                                                                                                                                                                                                                                                                                                                    |
| Interrupt Enable Register                         | The Interrupt Enable Register enables the interrupts reported by DMA Status Register. After a hardware or software reset, all interrupts are disabled.                                                                                                                                                                                                                                                                                                                                                                     |
| Missed Frame and Buffer Overflow Counter Register | The DMA maintains two counters to track the number of missed frames during reception. This Register reports the current value of the counter, which is used for diagnostic purposes.                                                                                                                                                                                                                                                                                                                                       |



**Table 23-8:** Summary of DMA related Registers. (Continued)

| Register Name                                 | Related Functionality                                                                                                                                          |
|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Receive Interrupt Watchdog Timer Register     | This register, when written with non-zero value, enables the watchdog timer for Receive Interrupt (RI) in DMA Status Register.                                 |
| SCB Bus Mode Register                         | The SCB Bus Mode Register controls the SCB interface master behavior. It is mainly used to control the burst splitting and the number of outstanding requests. |
| SCB Status Register                           | The active status of the SCB interface read & write channels are given in this Register.                                                                       |
| Current Host Transmit Descriptor Register     | The Current Host Transmit Descriptor Register points to the start address of the current Transmit Descriptor read by the DMA.                                  |
| Current Host Receive Descriptor Register      | The Current Host Receive Descriptor Register points to the start address of the current Receive Descriptor read by the DMA.                                    |
| Current Host Transmit Buffer Address Register | The Current Host Transmit Buffer Address Register points to the current Transmit Buffer Address being read by the DMA.                                         |
| Current Host Receive Buffer Address Register  | The Current Host Receive Buffer Address Register points to the current Receive Buffer address being read by the DMA.                                           |

**NOTE:** There should not be any further writes to the below registers until the first write is updated. Otherwise, the second write operation will not get updated properly. For correct operation, the delay between two writes to the same register location should be at least 8 cycles of 50 MHz RMII REFCLK.

**Table 23-9:** DMA Registers with Consecutive Writes

| Registers with Implications for Consecutive Writes |
|----------------------------------------------------|
| DMA Bus Mode Register                              |

## DMA Descriptors

The DMA module in the Ethernet subsystem transfers data based on a linked list of descriptors. The descriptor addresses must be aligned to the 32-bit bus width. The descriptors can be either 4 x 32-bit words (16 bytes) or 8 x 32-bit words (32 bytes). The controller needs to be configured for the appropriate WORD length using the DMA Bus Mode Register. Descriptor words are numbered from 0 to 7 for both transmit and receive engine.

Typical factors for deciding the descriptor word size:

1. When Time-stamping or receive checksum engine are not enabled, the extended descriptors are not required and the software can use descriptors with the default size of 16 bytes (4 words).
2. When Time-stamping feature is enabled (to be used with the IEEE 1588 PTP Engine), the software needs to allocate 32-bytes (8 words) of memory for every descriptor.
3. When only the receive checksum off-load is enabled (time stamping disabled) the software needs to allocate 32-bytes (8 words) of memory for every descriptor, although in reality only word 4 of the extended words (descriptors 4–7) contain the required status information. The rest of extended words may be treated as reserved or dummy.

## Transmit Descriptor

The transmit descriptor structure is shown in the below figure. The application software must program the TDES0 control bits during descriptor initialization. When the DMA updates the descriptor, it write backs all the control bits except the `OWN` bit (which it clears) and updates the status bits. The contents of the transmitter descriptor word 0 (TDES0) through word 7 (TDES7) are given in the following tables.

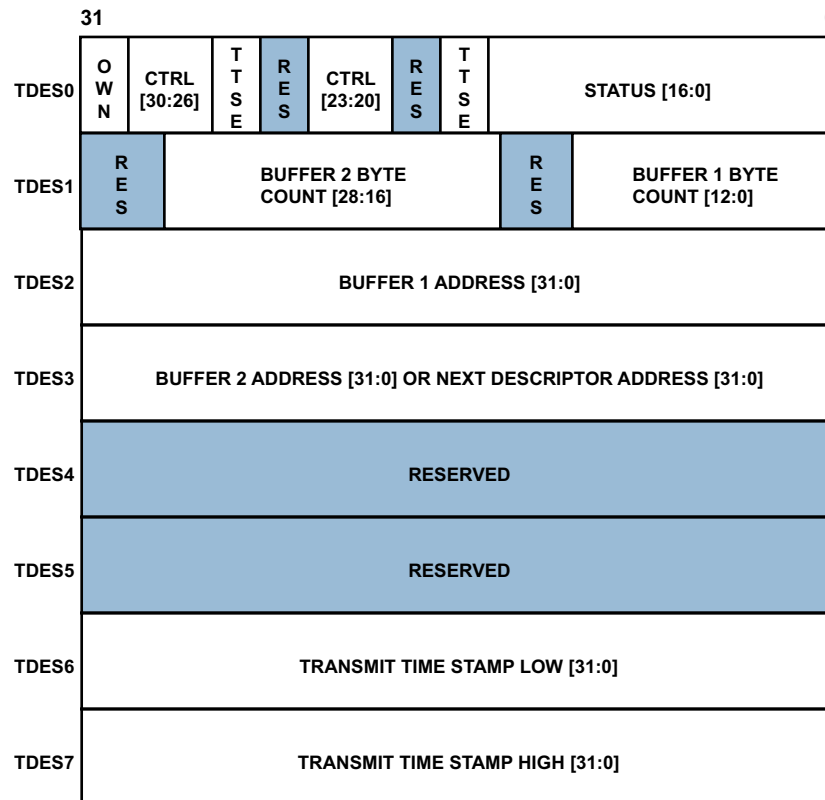


**Figure 23-7:** Transmit Descriptor words

**Table 23-10:** Transmit Descriptor Fields (TDES0)

| Bit | Name | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31  | OWN  | Own Bit. When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the application. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit. |
| 30  | IC   | Interrupt on Completion. When set, this bit sets the transmit interrupt (DMA status register [0]) after the present frame has been transmitted.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 29  | LS   | Last Segment. When set, this bit indicates that the buffer contains the last segment of the frame                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 28  | FS   | First Segment. When set, this bit indicates that the buffer contains the first segment of a frame.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 27  | DC   | Disable CRC. When this bit is set, the EMAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.                                                                                                                                                                                                                                                                                                                                                                           |
| 26  | DP   | Disable Pad. When set, the EMAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.                                                                                                                                                                                                                          |
| 25  | TTSE | Transmit Time Stamp Enable. When set, this bit enables IEEE1588 hardware time stamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.                                                                                                                                                                                                                                                                                                                                              |
| 24  |      | Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 23-10:** Transmit Descriptor Fields (TDES0) (Continued)

| Bit   | Name     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:22 | CIC      | Checksum Insertion Control. These bits control the checksum calculation and insertion. Bit encodings are as shown below.<br>2'b00 = Checksum Insertion Disabled.<br>2'b01 = Only IP header checksum calculation and insertion are enabled.<br>2'b10 = IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware.<br>2'b11 = IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware.                                                                                                 |
| 21    | TER      | Transmit End of Ring. When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 20    | TCH      | Second Address Chained. When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a <i>don't care</i> value. TDES0[21] takes precedence over TDES0[20].                                                                                                                                                                                                                                                                                                                                                       |
| 19:18 | Reserved |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 17    | TTSS     | Transmit Time Stamp Status. This field is used as a status bit to indicate that a time stamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a time stamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment control bit (TDES0[29]) is set.                                                                                                                                                                                                                                                                                                         |
| 16    | IHE      | IP Header Error. When set, this bit indicates that the EMAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5. |

**Table 23-10:** Transmit Descriptor Fields (TDES0) (Continued)

| Bit | Name | Description                                                                                                                                                                                                                                                                                                                                                                   |
|-----|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15  | ES   | Error Summary. Indicates the logical OR of the following bits.<br>TDES0[14] = Jabber Timeout<br>TDES0[13] = Frame Flush<br>TDES0[11] = Loss of Carrier<br>TDES0[10] = No Carrier<br>TDES0[9] = Late Collision<br>TDES0[8] = Excessive Collision<br>TDES0[2] = Excessive Deferral<br>TDES0[1] = Underflow Error<br>TDES0[16] = IP Header Error<br>TDES0[12] = IP Payload Error |
| 14  | JT   | Jabber Timeout. When set, this bit indicates the EMAC transmitter has experienced a jabber time-out. This bit is only set when the EMAC configuration register's JD bit is not set.                                                                                                                                                                                           |
| 13  | FF   | Frame Flushed. When set, this bit indicates that the DMA/MFL flushed the frame due to a software Flush command given by the CPU.                                                                                                                                                                                                                                              |
| 12  | IPE  | IP Payload Error. When set, this bit indicates that EMAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.                |
| 11  | LC   | Loss of Carrier. When set, this bit indicates that a loss of carrier occurred during frame transmission. This is valid only for the frames transmitted without collision when the EMAC operates in Half-Duplex mode.                                                                                                                                                          |
| 10  | NC   | No Carrier. When set, this bit indicates that the Carrier Sense signal from the PHY was not asserted during transmission.                                                                                                                                                                                                                                                     |
| 9   | LC   | Late Collision. When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte-times, including preamble). This bit is not valid if the Underflow Error bit is set.                                                                                                                                        |
| 8   | EC   | Excessive Collision. When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If the DR (Disable Retry) bit in the EMAC Configuration register is set, this bit is set after the first collision, and the transmission of the frame is aborted.                                          |

**Table 23-10:** Transmit Descriptor Fields (TDES0) (Continued)

| Bit | Name             | Description                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7   | VF               | VLAN Frame. When set, this bit indicates that the transmitted frame was a VLAN-type frame.                                                                                                                                                                                                                                                                                                  |
| 6:3 | CC               | Collision Count. This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.                                                                                                                                                                                        |
| 2   | ED               | Excessive Deferral? When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbps mode or if Jumbo Frame is enabled) if the Deferral Check (DC) bit in the EMAC Control register is set high.                                                                                                       |
| 1   | UF               | Underflow Error. When set, this bit indicates that the EMAC aborted the frame because data arrived late from the application memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]). |
| 0   | DB: Deferred Bit | When set, this bit indicates that the EMAC defers before transmission because of the presence of carrier. This bit is valid only in Half-Duplex mode.                                                                                                                                                                                                                                       |

**Table 23-11:** Transmit Descriptor Word 1 (TDES1)

| Bit   | Name     | Description                                                                                                                                                                                                                 |
|-------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–29 | Reserved |                                                                                                                                                                                                                             |
| 28–16 | TBS2     | Transmit Buffer 2 Size. These bits indicate the second data buffer size in bytes. This field is not valid if TDES0[20] is set.                                                                                              |
| 15–13 | Reserved |                                                                                                                                                                                                                             |
| 12–0  | TBS1     | Transmit Buffer 1 Size. These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]). |

**Table 23-12:** Transmit Descriptor 2 (TDES2)

| Bit  | Name                     | Description                                                                                                  |
|------|--------------------------|--------------------------------------------------------------------------------------------------------------|
| 31–0 | Buffer 1 Address Pointer | These bits indicate the physical address of buffer 1. There is no limitation on the buffer address alignment |

**Table 23-13:** Transmit Descriptor 3 (TDES3)

| Bit  | Name                                               | Description                                                                                                                                                                                                                                                                                                                                                      |
|------|----------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | Buffer 2 Address Pointer (Next Descriptor Address) | Indicates the physical address of buffer 2 when a descriptor ring structure is used. If the second address chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the next descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.) |

**Table 23-14:** Transmit Descriptor 6 (TDES6)

| Bit  | Name | Description                                                                                                                                                                                                                                                                                       |
|------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | TTSL | Transmit Frame Time Stamp Low. This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding transmit frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set. |

**Table 23-15:** Transmit Descriptor 7 (TDES7)

| Bit  | Name | Description                                                                                                                                                                                                                                                                                      |
|------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | TTSH | Transmit Frame Time Stamp High. This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field has the time stamp only if the Last Segment bit (LS) in the descriptor is set and Time stamp status (TTSS) bit is set. |

## Receive Descriptor

The structure of the receive descriptor is shown below. This can have 32 bytes of descriptor data (8 DWORDs) when advanced time stamping or checksum is enabled.

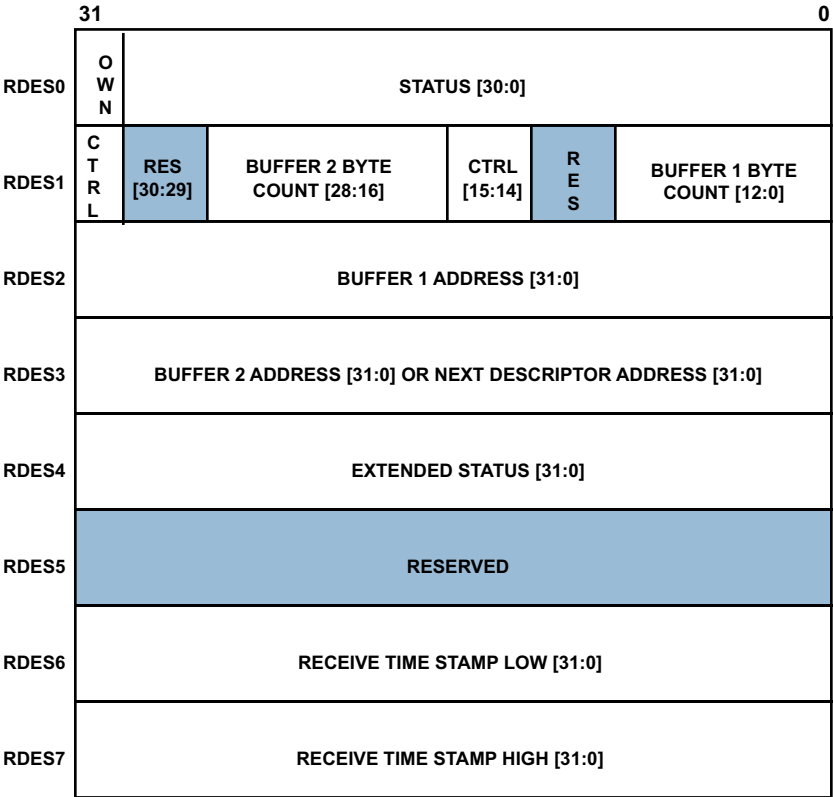


Figure 23-8: Receive Descriptor words

Table 23-16: Receive Descriptor Fields (RDES0)

| Bit   | Name | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31    | OWN  | Own. When set, this bit indicates that the descriptor is owned by the DMA of the EMAC Subsystem. When this bit is reset, this bit indicates that the descriptor is owned by the application. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.                                                                                                                                                                     |
| 30    | AFM  | Destination Address Filter Fail. When set, this bit indicates a frame that failed in the DA Filter in the EMAC CORE.                                                                                                                                                                                                                                                                                                                                                                                         |
| 29–16 | FL   | Frame Length. These bits indicate the byte length of the received frame that was transferred to application memory (including CRC). This field is valid when last descriptor (RDES0[8]) is set and either the descriptor error (RDES0[14]) or overflow error bits are reset. This field is valid when Last Descriptor (RDES0[8]) is set. When the last descriptor and error summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame. |



**Table 23-16:** Receive Descriptor Fields (RDES0) (Continued)

| Bit | Name                 | Description                                                                                                                                                                                                                                                                                                                                              |
|-----|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15  | ES                   | Error Summary. Indicates the logical OR of the following bits.<br>RDES0[1] = CRC Error<br>RDES0[4] = Watchdog Timeout<br>RDES0[6] = Late Collision<br>RDES0[7] = Giant Frame<br>RDES0[4:3] = IP Header/Payload Error<br>RDES0[11] = Overflow Error<br>RDES0[14] = Descriptor Error. This field is valid only when the last descriptor (RDES0[8]) is set. |
| 14  | DE                   | Descriptor Error. When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the next descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.                                                             |
| 13  | Reserved             |                                                                                                                                                                                                                                                                                                                                                          |
| 12  | LE                   | Length Error. When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.                                                                                                                                              |
| 11  | OE                   | Overflow Error. When set, this bit indicates that the received frame was damaged due to buffer overflow in MFL.                                                                                                                                                                                                                                          |
| 10  | VLAN                 | VLAN Tag. When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the EMAC CORE.                                                                                                                                                                                                                             |
| 9   | FS                   | First Descriptor. When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.                                            |
| 8   | LS                   | Last Descriptor. When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame                                                                                                                                                                                                                           |
| 7   | Time Stamp Available | When set, this bit indicates that a snapshot of the timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set                                                                                                                                                                  |
| 6   | LC                   | Late Collision. When set, this bit indicates that a late collision has occurred while receiving the frame in Half-Duplex mode.                                                                                                                                                                                                                           |

**Table 23-16:** Receive Descriptor Fields (RDES0) (Continued)

| Bit | Name                      | Description                                                                                                                                                                                                                                                                                  |
|-----|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5   | FT                        | Frame Type. When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 16'h0600). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames less than 14 bytes. |
| 4   | RWT                       | Receive Watchdog Timeout. When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.                                                                                          |
| 3   | Reserved                  |                                                                                                                                                                                                                                                                                              |
| 2   | DE                        | Dribble Bit Error. When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles).                                                                                                                                                                   |
| 1   | CE                        | CRC Error. When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.                                                                                                      |
| 0   | Extended Status Available | When set, this bit indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set.                                                                                                                          |

**Table 23-17:** Receive Descriptor Fields 1 (RDES1)

| Bit   | Name     | Description                                                                                                                                                                                                                                                                                                                                                                    |
|-------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31    | DIC      | Disable Interrupt on Completion. When set, this bit prevents setting the status register's RI bit (DMA status register [6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Application due to RI for that frame.                                                                          |
| 30–29 | Reserved |                                                                                                                                                                                                                                                                                                                                                                                |
| 28–16 | RBS2     | Receive Buffer 2 Size. These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, (32-bit bus), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set. |
| 15    | RER      | Receive End of Ring. When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.                                                                                                                                                                                          |

**Table 23-17:** Receive Descriptor Fields 1 (RDES1) (Continued)

| Bit  | Name     | Description                                                                                                                                                                                                                                                                                                                                                                                                    |
|------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14   | RCH      | Second Address Chained. When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a <i>don't care</i> value. RDES1[15] takes precedence over RDES1[14].                                                                                                                        |
| 13   | Reserved |                                                                                                                                                                                                                                                                                                                                                                                                                |
| 12–0 | RBS1     | Receive Buffer 1 Size. Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4 (32-bit bus), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4 the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14). |

**Table 23-18:** Receive Descriptor Fields 2 (RDES2)

| Bit  | Name                     | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | Buffer 1 Address Pointer | These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. Note that the DMA performs a write operation with the RDES2[1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[1:0] (corresponding to bus width of 32) if the address pointer is to a buffer where the middle or last part of the frame is stored. |

**Table 23-19:** Receive Descriptor Fields 3 (RDES3)

| Bit  | Name                                               | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|------|----------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | Buffer 2 Address Pointer (Next Descriptor Address) | These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[1:0] = 0, corresponding to a bus width of 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[1:0] (corresponding to a bus width of 32) if the address pointer is to a buffer where the middle or last part of the frame is stored. |

**Table 23-20:** Receive Descriptor Fields 4 (RDES4)

| Bit   | Name           | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–14 | Reserved       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 13    | PTP Version    | When set, indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format. This is valid only if the message type is non-zero.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 12    | PTP Frame Type | When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information on IPv4 or IPv6 can be obtained from bits 6 and 7.                                                                                                                                                                                                                                                                                                                     |
| 11–8  | Message Type   | These bits are encoded to give the type of the message received.<br>0000 = No PTP message received<br>0001 = SYNC (all clock types)<br>0010 = Follow_Up (all clock types)<br>0011 = Delay_Req (all clock types)<br>0100 = Delay_Resp (all clock types)<br>0101 = Pdelay_Req (in peer-to-peer transparent clock) or Announce (in ordinary or boundary clock)<br>0110 = Pdelay_Resp (in peer-to-peer transparent clock) or Management (in ordinary or boundary clock)<br>0111 = Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) or Signaling (for ordinary or boundary clock)<br>1xxx - Reserved |

**Table 23-20:** Receive Descriptor Fields 4 (RDES4) (Continued)

| Bit | Name                 | Description                                                                                                                                                                                                                                                                                                                                                                                 |
|-----|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7   | IPv6 Packet Received | When set, this bit indicates that the received packet is an IPv6 packet.                                                                                                                                                                                                                                                                                                                    |
| 6   | IPv4 Packet Received | When set, this bit indicates that the received packet is an IPv4 packet.                                                                                                                                                                                                                                                                                                                    |
| 5   | IP Checksum Bypassed | When set, this bit indicates that the checksum off-load engine is bypassed.                                                                                                                                                                                                                                                                                                                 |
| 4   | IP Payload Error     | When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.                                                       |
| 3   | IP Header Error      | When set, this bit indicates either that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value.                                                                                                                                                                 |
| 2–0 | IP Payload Type      | These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Off-load Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload due to an IP header error or fragmented IP.<br>3'b000 = Unknown or did not process IP payload<br>3'b001 = UDP<br>3'b010 = TCP<br>3'b011 = ICMP<br>3'b1xx = Reserved |

**Table 23-21:** Receive Descriptor Fields 6 (RDES6)

| Bit  | Name | Description                                                                                                                                                                                                                                                                                               |
|------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | RTSL | Receive Frame Time Stamp Low. This field is updated by DMA with the least significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]). |

**Table 23-22:** Receive Descriptor Fields 7 (RDES7)

| Bit  | Name | Description                                                                                                                                                                                                                                                                                               |
|------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31–0 | RTSH | Receive Frame Time Stamp High. This field is updated by DMA with the most significant 32 bits of the time stamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]). |

## EMAC DMA Transmit Process

The following sections describe how the direct memory access transmit process works on the EMAC controller.

- [Default \(Non-OSF\) Mode](#)
- [OSF Mode Enabled](#)
- [Transmit Frame Processing](#)
- [Transmit Polling Suspended](#)

## Default (Non-OSF) Mode

The transmit DMA engine in default mode proceeds as follows:

1. The application sets up the transmit descriptor (using TDES0- TDES3) and sets the OWN bit (TDES0) after setting up the corresponding data buffer(s) with Ethernet frame data.
2. Once the ST bit (operation mode register [13]) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the transmit descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the application, or if an error condition occurs, transmission is suspended and both the transmit buffer unavailable (DMA\_STAT [2]) and normal interrupt summary (DMA\_STAT [16]) bits are set. The transmit engine proceeds to Step 9.
4. If the acquired descriptor is flagged as owned by DMA (TDES0 [31] = 1#b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the transmit data from the application memory and transfers the data to the MFL for transmission.
6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of-Ethernet-frame data is transferred to the MFL.
7. When frame transmission is complete, if IEEE 1588 time stamping was enabled for the frame (as indicated in the transmit status) the time-stamp value obtained from MFL is written to the transmit

descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the application now owns this descriptor. If time stamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3.

8. Transmit Interrupt (DMA\_STAT [0]) is set after completing transmission of a frame that has Interrupt on Completion (TDES1 [31]) set in its Last Descriptor. The DMA engine then returns to Step 3.
9. In the Suspend state, the DMA tries to re-acquire the descriptor (and thereby return to Step 3) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared.

**NOTE:** If the OSF bit is not set, the actual Inter Frame Gap (IFG) may be seen as more than as programmed in the MAC configuration register.

## OSF Mode Enabled

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first (if the OSF bit is set in the DMA\_OPMODE [2]). As the transmit process finishes transferring the first frame, it immediately polls the Transmit Descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame status information.

In OSF mode, the run state transmit DMA operates in the following sequence:

1. The DMA operates as described in steps 1-6 of the TxDMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into suspend mode and skips to Step 7.
4. The DMA fetches the transmit frame from the application memory and transfers the frame to the MFL until the End-of-Frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.
5. The DMA waits for the previous frame's frame transmission status and time stamp. Once the status is available, the DMA writes the time stamp to TDES2 and TDES3, if such time stamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared OWN bit, to the corresponding TDES0, thus closing the descriptor. If time stamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the transmit interrupt is set; the DMA fetches the next descriptor, and then proceeds to Step 3 (when status is normal). If the previous transmission status shows an underflow error, the DMA goes into suspend mode (Step 7).
7. In suspend mode, if a pending status and time stamp are received from the MFL, the DMA writes the time stamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to suspend mode.

8. The DMA can exit suspend mode and enter the run state (go to Step 1 or Step 2 depending on pending status) only after receiving a transmit poll demand (DMA\_TXPOLL).

**NOTE:** If OSF bit is set, the DMA fetches the next descriptor in advance before closing the current descriptor. Therefore the descriptor chain should have more than two different descriptors for proper operation.

**NOTE:** If the OSF bit is set, the DMA starts fetching the second frame immediately after completing the transfer of the first frame to the FIFO. It does not wait for the status to be updated. In the meantime the MFL receives the second frame into the FIFO while transmitting the first frame. The difference in cycles will not be seen for the first descriptor, because the time taken for the complete descriptor processing remains the same whether OSF is set or not. The difference is seen only for the following descriptor as the processing of that descriptor is started earlier.

## Transmit Frame Processing

The Transmit DMA Engine expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The destination address, source address, and type/length fields contain valid data. If the transmit descriptor indicates that the EMAC CORE must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the first descriptor (TDES1 [29]) and the last descriptor (TDES1 [30]), respectively.

As transmission starts, the first descriptor must have (TDES1 [29]) set. When this occurs, frame data transfers from the application buffer to the transmit FIFO. Concurrently, if the current frame has the Last Descriptor (TDES1 [30]) cleared, the transmit process attempts to acquire the next descriptor. The transmit process expects this descriptor to have TDES1 [29] clear. If TDES1 [30] is clear, it indicates an intermediary buffer. If TDES1 [30] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the transmit descriptor 0 (TDES0) word of the descriptor that has the last segment set in transmit descriptor 1 (TDES1 [30]). At this time, if interrupt on completion (TDES1 [31]) was set, transmit interrupt (DMA\_STAT [0]) is set, the next descriptor is fetched, and the process repeats.

Actual frame transmission begins after the MFL transmit FIFO has reached either a programmable transmit threshold (DMA\_OPMODE [16:14]), or a full frame is contained in the FIFO. There is also an option for store and forward mode (DMA\_OPMODE [21]). Descriptors are released (OWN bit TDES0 [31] clears) when the DMA finishes transferring the frame.

## Transmit Polling Suspended

Transmit polling may be suspended by either of the following conditions:

1. The DMA detects a descriptor owned by the application (TDES0 [31] =0).
2. A frame transmission is aborted when a transmit error due to underflow is detected. The appropriate transmit descriptor 0 (TDES0) bit is set.



If the second condition occurs, both abnormal interrupt summary ([15]) and transmit underflow bits ([5]) are set and the information is written to transmit descriptor 0, causing the suspension. If the DMA goes into SUSPEND state due to the first condition then both Normal Interrupt Summary ([16]) and Transmit Buffer Unavailable ([2]) are set.

In both cases, the position in the transmit list is retained. The retained position is that of the descriptor following the last descriptor closed by the DMA.

The driver must explicitly issue a transmit poll demand command after rectifying the suspension cause. If the first condition occurs, the driver must give descriptor ownership to the DMA and then issue a poll demand command, in order to resume the transfer.

## EMAC DMA Receive Process

The following sections describe how the direct memory access receive process works on the EMAC controller.

- [Receive Frame Processing](#)
- [Receive Descriptor Acquisition](#)
- [Receive Process Suspended](#)

The Receive DMA engine's reception proceeds as follows:

1. The application sets up Receive descriptors (RDES0-RDES3) and sets the OWN bit (RDES0 [31]).
2. Once the SR (DMA\_OPMODE [1]) bit is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the application), the DMA enters the Suspend state and jumps to Step 9.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to Step 7. If IEEE 1588 time stamping is enabled, the DMA writes the time stamp (if available) to the current de. If the DMA does not own the next fetched descriptor and the frame transfer is not complete, the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled). The DMA closes the current descriptor (clears the OWN bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to Step 8. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to Step 4.
7. If IEEE 1588 time stamping is enabled, the DMA writes the time stamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MFL and writes the status word to the current descriptor's RDES0, with the Own bit cleared and the last segment bit set.

8. The receive engine checks the latest descriptor's OWN bit. If the host owns the descriptor (OWN bit is 0) the receive buffer unavailable bit (DMA\_STAT [7]) is set and the DMA receive engine enters the Suspended state (Step 9). If the DMA owns the descriptor, the engine returns to Step 4 and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the receive FIFO (you can control flushing using Bit 24 of DMA\_OPMODE).
10. The receive DMA exits the suspend state when a receive poll demand is given or the start of next frame is available from the MFL's Receive FIFO. The engine proceeds to Step 2 and re fetches the next descriptor.

## Receive Frame Processing

The EMAC transfers the received frames to the application memory only when the frame passes the address filter sub-block and frame size is greater than or equal to configurable threshold bytes set for the receive FIFO of MFL, or when the complete frame is written to the FIFO in Store-and-Forward mode.

If the frame fails the address filtering, it is dropped in the EMAC block itself (unless receive all EMAC MACCFG[31] bit is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be purged from the Receive FIFO.

After 64 bytes (configurable threshold) have been received, the MFL block requests the DMA block to begin transferring the frame data to the Receive Buffer pointed to by the current descriptor. The DMA sets First Descriptor (RDES0 [9]) after the SCB becomes ready to receive the data (if DMA is not fetching transmit data from the application). The descriptors are released when the Own (RDES [31]) bit is reset to 0, either as the data buffer fills up or as the last segment of the frame is transferred to the receive buffer. If the frame is contained in a single descriptor, both last descriptor (RDES [8]) and first descriptor (RDES [9]) are set.

The DMA fetches the next descriptor, sets the last descriptor (RDES [8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets Receive Interrupt (DMA\_STAT [6]). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application. If this occurs, the receive process sets receive buffer unavailable (DMA\_STAT [7]) and then enters the suspend state. The position in the receive list is retained.

## Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- The receive Start/Stop bit (DMA\_OPMODE [1]) has been set immediately after being placed in the Run state.
- The data buffer of current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current receive descriptor is not yet closed.

- The receive process has been suspended because of an application-owned buffer (RDES0 [31] = 0) and a new frame is received.
- A receive poll demand has been issued.

## Receive Process Suspended

If a new receive frame arrives while the receive process is in suspend state, the DMA re-fetches the current descriptor in the application memory. If the descriptor is now owned by the DMA, the receive process re-enters the Run state and starts frame reception. If the descriptor is still owned by the application, by default, the DMA discards the current frame at the top of the receive FIFO and increments the missed frame counter. If more than one frame is stored in the receive FIFO, the process repeats.

The discarding or flushing of the frame at the top of the receive FIFO can be avoided by setting operation mode register bit 24 (DFF). In such conditions, the receive process sets the receive buffer unavailable status and returns to the Suspend state.

## OWN Bit (Ownership) Semaphore

Usage or ownership of the transmit/receive descriptor between application and EMAC is mutually exclusive. While the EMAC is accessing the descriptor, the application cannot modify it. Conversely, while the host is updating the descriptor, the EMAC cannot use the descriptor's contents. This functionality is implemented through the OWN bit in the transmit/receive descriptor, acting as a semaphore to prevent multiple, simultaneous access to the descriptors.

Following example is based on a use case of 4 WORDs enabled for descriptors (which means the ATDS bit in DMA\_BUSMODE register is not set) and Chain Structure configuration is assumed. However, the explanation of OWN bit semaphore remains consistent irrespective of any particular mode of operation.

### 1. Transmit Own Bit:

- TDES0-TDES3 words implement the transmit descriptors. TDES0 [31] is defined as the OWN bit. When TDES0 [31] is set to 0, this bit indicates that the descriptor is available for the application to update. The application sets up the descriptors, including the buffer addresses, by updating TDES0 through TDES3.
- To release ownership of the descriptor to the EMAC, the application sets the transmit Own bit, TDES0 [31], to 1. TDES0 [31] = 1 indicates that the descriptor is ready for use by the EMAC. The DMA reads the descriptors, then fetches the data to be transmitted from the buffer locations pointed to by the transmit descriptors (TDES2 and TDES3). When either the last data buffer is empty or the end-of-frame is reached, DMA clears the TDES0 [31] bit to 0. Now the transmit descriptor is released to the application for updates.

### 2. Receive Own Bit:

- RDES0-RDES3 words implement the receive descriptors. RDES0 [31] is defined as the Own bit. When RDES0 [31] is set to 0, this bit indicates that the descriptor is available for the application to update. The application sets up the descriptors, including the buffer locations for writing the

received data, by updating RDES0 through RDES3. To give ownership of the descriptor to the EMAC, the host sets the receive Own bit, RDES0 [31], to 1.

- RDES0 [31] = 1 indicates that the descriptor is ready for use by the EMAC. The EMAC's DMA reads the descriptors, then writes the received data to the buffers with locations pointed to by the receive descriptors (RDES2 and RDES3). When either the last data buffer is full or the end-of-frame is reached, DMA clears the RDES0 [31] bit to 0. Now the receive descriptor is released to the application for updates

## Application Data Buffer Alignment

The transmit and receive data buffers do not have any restrictions on start address alignment; the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates transfers with address aligned to the bus width with dummy data for the byte lanes not required. This typically happens during the transfer of the beginning or end of an Ethernet frame.

Example for Buffer Read—If the transmit buffer address is 0xFF800002 and 15 bytes need to be transferred, then the DMA reads 5 full words (5 x 32-bit data) from address 0xFF800000. However, when transferring data to the EMAC transmit FIFO, the extra bytes (the first two bytes) are dropped or ignored. Similarly, the last 3 bytes of the last transfer are also ignored. The DMA always ensures it transfers a full 32-bit data to the transmit FIFO, unless it is the end-of-frame.

Example for Buffer Write—If the receive buffer address is 0xFF800002 and 15 bytes of a received frame need to be transferred, then the DMA writes 5 full words (5 x 32-bit data) to address 0xFF800000. However, the first 2 bytes of first transfer and the last 3 bytes of the third transfer have dummy data.

## Buffer Size Calculations

The DMA Engines do not update the size fields in the Transmit and Receive descriptors by itself. The DMA updates only the status fields (RDES0 and TDES0) of the descriptors. The driver has to perform the size calculations. The transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the EMAC CORE. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the EMAC.

The receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MFL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

## EMAC FIFO Layer (EMAC MFL)

The MAC FIFO Layer provides FIFO memory to buffer and regulate the frames between the application system memory and the EMAC CORE. It also enables the data to be transferred between the application clock domain and the EMAC clock domains. The MFL layer has transfer controllers for each direction, namely the Transmit Controller (TxFIFO) and the Receive Controller (RxFIFO). The data path for both directions is 32-bit wide and each controller has a dedicated FIFO.

### FIFO Size

- The transmit FIFO size is fixed and is 256 bytes.
- The receive FIFO sized is fixed and is 128 bytes.

### FIFO Layer Transmit Path

The DMA Engine controls all transactions for the transmit path with the application. Ethernet frames read from the system memory are pushed into the FIFO by the DMA. The frame is then popped out and transferred to the EMAC CORE when triggered. When the end-of-frame is transferred, the status of the transmission is taken from the EMAC CORE and transferred back to the DMA. FIFO-fill level is indicated to the DMA so that it can initiate a data fetch in required bursts from the system memory, through the SCB interface.

When Operate on Second Frame (OSF) is enabled, the MFL receives the second frame into the FIFO while transmitting the first frame. As soon as the first frame has been transferred and the status is sent to DMA. If the DMA has already completed sending the second packet to the MFL, it must wait for the status of the first packet before proceeding to the next frame.

Modes of operation for FIFO transactions:

1. Threshold mode: In this mode as soon as the number of bytes in the FIFO crosses the configured threshold level (or when the end-of-frame is written before the threshold is crossed), the data is ready to be popped out and forwarded to the EMAC core. The threshold level is configured using the TTC bits of the DMA bus mode register.
2. Store-and-Forward mode: In this mode, the MFL pops the frame towards the EMAC core only after a complete frame is stored in the FIFO. If the Tx FIFO size is smaller than the Ethernet frame to be transmitted (such as a Jumbo frame), then the frame is forwarded when the Tx FIFO becomes almost full or when the requested FIFO does not have space to accommodate the requested burst-length. Therefore, the FIFO read controller never stalls in Store and Forward mode even if the Ethernet frame length is bigger than the Tx FIFO depth.

### Transmit FIFO and Half-Duplex Retransmissions

While a frame is being transferred from the FIFO a collision event can occur on the EMAC line interface in Half-Duplex mode. The EMAC then indicates a retry attempt to the MFL by giving the status even before the end-of-frame is transferred from MFL. Then the MFL will enable the retransmission by popping out the frame again from the FIFO.

After more than 96 bytes are popped out of FIFO, the FIFO controller frees up that space and makes it available to the DMA to push in more data. This means that the retransmission is not possible after this threshold is crossed or when the EMAC CORE indicates a late-collision event.

### Transmit FIFO Flush Operation

The EMAC provides a control to the software to flush the Transmit FIFO in the MFL layer through the use of Bit 20 of the Operation Mode register. The Flush operation is immediate and the MFL clears the Transmit FIFO and the corresponding pointers to the initial state even if it is in the middle of transferring a frame to the EMAC CORE. The data which is already accepted by the MAC transmitter will not be flushed. It will be scheduled for transmission and will result in underflow as Transmit FIFO does not complete the transfer of rest of the frame. As in all underflow conditions, a runt frame will be transmitted and observed on the line. The status of such a frame will be marked with both Underflow and Frame Flush events (TDES0 bits 13 and 1).

The MFL layer also stops accepting any data from the application (DMA) during the flush operation. It generates and transfers transmit status words to the application for the number of frames that is flushed inside the MFL (including partial frames). Frames that are completely flushed in the MFL have the Frame Flush Status bit (TDES0 bit 13) set. The MFL completes the flush operation when the application (DMA) accepts all of the status words for the frames that were flushed, and then clears the Transmit FIFO Flush control register bit. At this point, the MFL starts accepting new frames from the application (DMA).

### FIFO Layer Receive Path

The receive controller operates in below sequence:

1. When the EMAC core receives a frame, it pushes in data with the frame start and end indicators. The MFL accepts the data and pushes it into the FIFO.
2. The receive controller takes the data out of the FIFO and sends it to the DMA.
  - In the default Threshold mode, when 64 bytes (configured with `RTC` bits of DMA Operation Mode register) or a full packet of data are received into the FIFO, the receive controller pops out the data and indicates its availability to the DMA. Some error frames may not be dropped, because the error status is received at the end-of-frame, by which time the start of that frame has already been read out of the FIFO.
  - In Rx FIFO Store-and-Forward mode (configured by the `RSF` bit of DMA Operation Mode register), a frame is read out only after being written completely into the Receive FIFO. In this mode, all error frames are dropped (if the EMAC core is configured to do so) such that only valid frames are read out and forwarded to the application.
3. After the end-of-frame is transferred, the status word from EMAC core is also the pushed FIFO. When the status of a partial frame due to overflow is given out, the Frame Length field in the status word is not valid.

## Receive FIFO Multi-Frame Handling

Since the status is available immediately following the data, the MFL is capable of storing any number of frames into the FIFO, as long as it is not full.

## Receive FIFO Error Handling

If the MFL Rx FIFO is full before it receives the end-of-frame data from the EMAC, an overflow is declared, the whole frame (including the status word) is dropped, and the overflow counter in the DMA (Over Flow Counter register) is incremented. This is true even if the Forward Error Frame (FEF bit of DMA Operation Mode register) is set. If the start address of such a frame has already been transferred, the rest of the frame is dropped and a dummy end-of-frame is written to the FIFO along with the status word. The status will indicate a partial frame due to overflow. In such frames, the Frame Length field is invalid.

The MFL Rx control logic can filter error and undersized frames, if enabled (using the DMA Operation Mode register FEF and FUF bits). If the start address of such a frame has already been transferred to the Rx FIFO read controller, that frame is not filtered. The start address of the frame is transferred to the Read Controller after the frame crosses the receive threshold (set by the DMA Operation Mode register RTC bits).

If the MFL Receive FIFO is configured to operate in Store-and-Forward mode, all error frames can be filtered and dropped.

## EMAC CORE

EMAC CORE is the lowest block in the EMAC peripheral and it performs all operations with the external world (PHY chip). It has independent transmit and receive modules that interact with the EMAC FIFO Layer at one end and the PHY chip via RMII interface at the other end. Both the modules have several sub blocks, and are explained in further sections.

Transmission is initiated when the MFL (FIFO Layer) pushes in data with start-of-frame and the CORE subsequently transmitting to the RMII. After the end-of-frame is transferred out, it gives out the status of transmission back to the MFL to be forwarded to application via DMA. A receive operation is initiated when the EMAC detects an SFD on the RMII. The CORE strips the preamble and SFD before proceeding to process the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC for the frame. The frame is dropped in the core if it fails the address filter.

**NOTE:** The “CORE” terminology (written in capitals) is used refer to the internal block of Ethernet peripheral, and should not be confused as “processor core”.

**Table 23-23:** EMAC CORE Related Registers

| Register name                | Related Functionality                                                                                                                                                                                                                                                                                                                                                                            |
|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAC Configuration Register   | Establishes receive and transmit operating modes. Important configurations include: <ul style="list-style-type: none"> <li>• Watchdog/Jabber/Jumbo frame sizes.</li> <li>• Inter Frame Gap.</li> <li>• Speed Control – 10/100 Mbps.</li> <li>• Full/Half Duplex.</li> <li>• Loopback Mode.</li> <li>• Checksum Offload.</li> <li>• Enabling TX/RX Engines.</li> </ul>                            |
| MAC Frame Filter Register    | Contains the filter controls for receiving frames. Some of the controls from this register go to the address check block of the MAC, which performs the first level of address filtering. The second level of filtering is performed on the incoming frame, based on other controls such as Pass Bad Frames and Pass Control Frames.                                                             |
| Hash Table High/Low Register | A 64-bit Hash table is used for group address filtering. For hash filtering, the contents of the destination address in the incoming frame is passed through the CRC logic, and the upper 6 bits of the CRC register are used to index the contents of the Hash table.                                                                                                                           |
| SMI Address Register         | Controls the management cycles to the external PHY through the Station Management interface. The register also includes field to program the frequency of MDC.                                                                                                                                                                                                                                   |
| SMI Data Register            | Stores write data to be written to the PHY register located at the address specified in SMI Address register. This register also stores read data from the PHY register located at the address specified by SMI Address Register.                                                                                                                                                                |
| Flow Control Register        | Controls the generation and reception of the control (pause command) frames by the EMAC's flow control module. The fields of the control frame are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Host must make sure that the Activate bit is cleared before writing to the register. |



**Table 23-23:** EMAC CORE Related Registers (Continued)

| Register name                  | Related Functionality                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| VLAN Tag Register              | Contains the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the receiving frame (Length/Type) with 16.h8100, and the following 2 bytes are compared with the VLAN tag; if a match occurs, it sets the received VLAN bit in the receive frame status. The legal length of the frame is increased from 1518 bytes to 1522 bytes.                  |
| Debug Register                 | Provides the status of all main modules of the transmit and receive data-paths and the FIFOs. An all-zero status indicates that the MAC core is in idle state (and FIFOs are empty) and no activity is going on in the data-paths.                                                                                                                                                             |
| Interrupt Status Register      | The contents of this register identify the events in the EMAC-CORE that can generate interrupts – MMC and PTP related.                                                                                                                                                                                                                                                                         |
| Interrupt Mask Register        | Enables the program to mask the interrupt signal because of the corresponding PTP event in the Interrupt Status register.                                                                                                                                                                                                                                                                      |
| MAC Address0 High/Low Register | Holds the upper/lower 16 bits of the MAC address of the station. Note that the first DA byte that is received on the RMII interface corresponds to the LS Byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 is the first byte) on the RMII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211. |

**NOTE:** Please refer to the Register Description for the detailed bit-level explanation of the Registers.

**NOTE:** There should not be any further writes to the below registers until the first write is updated. Otherwise, the second write operation will not get updated properly. For correct operation, the delay between two writes to the same register location should be at least 8 cycles of 50MHz RMII REFCLK.

**Table 23-24:** MAC Registers with Consecutive Write Implications

| Registers with Implications for Consecutive Writes |
|----------------------------------------------------|
| Operation Mode Register                            |
| MAC Configuration register                         |
| Hash Table High Register                           |
| Hash Table Low Register                            |

**Table 23-24:** MAC Registers with Consecutive Write Implications (Continued)

| Registers with Implications for Consecutive Writes |
|----------------------------------------------------|
| SMI Address Register                               |
| SMI Data Register                                  |
| Flow Control Register                              |
| VLAN Tag Register                                  |
| MAC Address High Register                          |
| MAC Address Low Register                           |

## EMAC CORE Transmission Engine

The following modules constitute the transmission function (transmission engine components) of the EMAC:

- [Transmit Bus Interface Module \(TBU\)](#)
- [Transmit Frame Controller Module \(TFC\)](#)
- [Transmit Checksum Offload Engine \(TCOE\)](#)
- [Transmit Protocol Engine Module \(TPE\)](#)
- [Transmit Scheduler Module \(STX\)](#)
- [Transmit CRC Generator Module \(CTX\)](#)
- [Transmit Flow Control Module \(FTX\)](#)

### Transmit Bus Interface Module (TBU)

This module interfaces the transmit path of the EMAC CORE with the MAC Layer FIFO interface. This module outputs the Transmit Status to the application at the end of normal transmission or collision.

### Transmit Frame Controller Module (TFC)

The Transmit Frame Controller regulates frames as well as converts the 32-bit input data into an 8-bit stream.

When the number of bytes received from the application falls below 60 (DA+SA+LT+DATA), the state machine automatically appends zeros to the transmitting frame to make the data length exactly 46 bytes to meet the minimum data field requirement of IEEE 802.3. The EMAC can also be programmed not to append any padding.

The Frame Controller receives the computed CRC and appends it as the FCS field to the data being transmitted out. When the EMAC is programmed to not append the CRC value to the end of Ethernet frames, the TFC module ignores the computed CRC. An exception to this rule is that when the EMAC is programmed to append pads for frames (DA+SA+LT+DATA) less than 60 bytes, then the CRC is always appended at the end of padded frame.

### Transmit Checksum Offload Engine (TCOE)

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the EMAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path.

*The checksum for TCP, UDP, or ICMP is calculated over a complete frame, and then inserted into its corresponding header field. Because of this requirement, this function is enabled only when the Transmit FIFO is configured for Store-and-Forward mode (that is, when the TSF bit is set in DMA Opmode Register. If the MAC is configured for Threshold (cut-through) mode, the Transmit COE is bypassed.*

*You must make sure that the Transmit FIFO is deep enough to store a complete frame before that frame is transferred to the EMAC CORE transmitter. User must enable the checksum insertion only in the frames that are less than the following number of bytes in size (even in the store-and-forward mode): FIFO Depth (256 bytes) – PBL – 3 FIFO Locations, where PBL is the programmed burst-length in DMA Bus mode register.*

### IP Header Checksum

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in transmit descriptor word TDES0. This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

For IPv4 datagrams:

- The received Ethernet type is 0x0800, but the IP header's Version field is not equal to 0x4.
- The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
- The total frame length is less than the value given in the IPv4 Header Length field.

For IPv6 datagrams:

- The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.
- The frame ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

*IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.*

## TCP/UDP/ICMP Checksum

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP.

*See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.*

*For non-TCP/UDP/ICMP/ICMPv6 payloads, this checksum engine is bypassed and nothing further is modified in the frame.*

*For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.*

*Fragmented IP frames (IPv4 or IPv6), IP frames with security features (such as an encapsulated security payload), and IPv6 frames with routing headers are not processed by this engine. The checksum engine bypasses the checksum insertion for such frames even if the checksum insertion is enabled.*

The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. This engine can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input frame. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Descriptor word TDES0. This engine sets the Payload Checksum Error status bit when it detects that the frame has been forwarded to the MAC Transmitter engine in the Store-and-Forward mode without the end-of-frame (EOF) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Transmit Checksum offloading is enabled by setting the `CIC` field bits [23:22] of TDES0 word in Transmit descriptor.

## Transmit Protocol Engine Module (TPE)

The Transmit Protocol Engine consists of a state-machine that controls the protocol level operation of Ethernet frame transmission. The module performs the following functions to meet the IEEE 802.3 specifications.

- Generates preamble and SFD.
- Generates jam pattern in Half-Duplex mode.
- Jabber timeout.
- Flow control for Half-Duplex mode (back pressure).
- Generates transmit frame status.

When a new frame transmission is requested, the Protocol Engine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of '10101010' pattern and the SFD is defined as 1 byte of '10101011' pattern.

The collision window is defined as 1 slot time (512 bit times for 10/100 Mbps). The jam pattern generation is applicable only to Half-Duplex mode, not to Full-Duplex mode. If a collision occurs any time from the beginning of the frame to the end of the CRC field, the state machine sends a 32-bit jam pattern of 0x55555555 on the RMII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, it completes the transmission of preamble and SFD and then sends the jam pattern. If the collision occurs after the collision window and before the end of the FCS field, it sends a 32-bit jam pattern and sets the late collision bit in the transmit frame status.

The module maintains a jabber timer (in 10/100-Mbps) to cut off the transmission of Ethernet frames if the TFC module transfers more than 2,048 (default) bytes. The time-out is changed to 10,240 bytes when the Jumbo frame is enabled.

The Transmit state machine uses the deferral mechanism for the flow control (Back Pressure) in Half-Duplex mode. When the application requests to stop receiving frames, the module sends a JAM pattern of 32 bytes whenever it senses a reception of a frame, provided the transmit flow control is enabled. This will result in a collision and the remote station will back off. The application can request a flow control signal by setting BPA bit of EMAC Flow Control Register. If the application requests a frame to be transmitted, then it will be scheduled and transmitted even when the backpressure is activated. Note that if the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur) then the remote stations will abort their transmissions due to excessive collisions.

## Transmit Scheduler Module (STX)

The Transmit Scheduler is responsible for scheduling the frame transmission on the RMII. The two major functions of this module are:

- Maintain the inter-frame gap between two transmitted frames.
- Follow the Truncated Binary Exponential Back-off algorithm for Half-Duplex mode.

The Scheduler maintains an idle period of the configured inter-frame gap (IFG bits of MAC Config Register) between any two transmitted frames. The Scheduler starts its IFG counter as soon as the carrier signal of the RMII goes inactive. In Half-Duplex mode and when IFG is configured for 96 bit times, the Scheduler follows the rule of deference specified in Section 4.2.3.2.1 of the IEEE 802.3 specification. The module resets its IFG counter if a carrier is detected during the first two-thirds (64-bit times for all IFG values) of the IFG interval. If the carrier is detected during the final one third of the IFG interval, the Scheduler continues the IFG count and enables the transmitter after the IFG interval.

### Transmit CRC Generator Module (CTX)

The Transmit CRC Generator module generates CRC for the FCS field of the Ethernet frame (DA + SA + LT + DATA + PAD).

This module calculates the 32-bit CRC for the FCS field of the Ethernet frame. The encoding is defined by the following polynomial:  $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ .

### Transmit Flow Control Module (FTX)

The Transmit Flow Control module generates Pause frames and transmits them to the Frame Controller as necessary, in Full-Duplex mode. The application can request the Flow Control module to send a Pause frame by setting the `FCB` bit in the Flow Control register.

If the application has requested for flow control, the Flow Control module will generate and transmit a single Pause frame. The value of the Pause Time in the generated frame contains the programmed Pause Time value in Flow Control Register. To extend the pause or end the pause prior to the time specified in the previously transmitted Pause frame, the application must request another Pause frame transmission after programming the Pause Time register with appropriate value.

If the flow control signal goes inactive prior to the sampling time, the Flow Control module will transmit a Pause frame with zero Pause Time to indicate to the remote end that the receive buffer is ready to receive new data frames.

### EMAC CORE Reception Engine

The following are the functional blocks (reception engine components) in the Receive path of the EMAC core.

- [Receive Protocol Engine Module \(RPE\)](#)
- [Receive CRC Module \(CRX\)](#)
- [Receive Frame Controller Module \(RFC\)](#)
- [Receive Flow Control Module \(FRX\)](#)
- [Receive Checksum Offload Engine \(RCOE\)](#)
- [Receive Bus Interface Unit Module \(RBU\)](#)

- [Address Filtering Module \(AFM\)](#)

## Receive Protocol Engine Module (RPE)

The Protocol Engine consists of the state-machine which strips the incoming preamble and SFD. Once the RXDV signal of the RMII becomes active, the Protocol Engine begins hunting for the SFD field from the receive modifier logic. Until then, the state machine drops the receiving preambles. Once the SFD is detected, it begins sending the data of the Ethernet frame to the Frame Controller, beginning with the first byte following the SFD (Destination Address).

**NOTE:** According to the IEEE 802.3 Ethernet Specifications, the EMAC receiver need not look or check for the preamble pattern. It has to wait only for the SFD pattern to identify the start of a frame. Hence the EMAC receiver will accept a frame even when no preamble is received before the SFD pattern.

The Protocol Engine also decodes the Length/Type field of the receiving Ethernet frame. If the Length/Type field is less than 0x600 and if the MAC is programmed for the auto CRC/PAD stripping option, the state machine sends the data of the frame up to the count specified in the Length/Type field, then starts dropping bytes (including the FCS field).

If the Length/Type field is greater than or equal to 0x600, the Protocol Engine will send all received Ethernet frame data to the Frame Controller, irrespective of the value on the programmed auto-CRC strip option.

As a default, the EMAC is programmed for watchdog timer to be enabled, that is, frames above 2,048 (10,240 if Jumbo Frame is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the Protocol Engine. This feature can be disabled by programming the Watchdog Disable in EMAC Configuration register. However even if the watchdog timer is disabled, frames greater than 16 KB in size are cut off and a watchdog time-out status is given.

The EMAC supports loopback of transmitted frames onto its receiver. As a default, the EMAC loopback function is disabled, but this feature can be enabled by programming the EMAC Configuration register, Loopback bit.

At the end of every received frame, the Protocol Engine generates received frame status and sends it to the Frame Controller. Control, missed frame, and filter fail status are added to the receive status in the Frame Controller.

## Receive CRC Module (CRX)

The Receive CRC Module checks for any CRC error in the receiving frame.

This module calculates the 32-bit CRC for the received frame that includes the Destination address field through the FCS field (DA+SA+LT+DATA+PAD+FCS). The encoding is defined by the following generating polynomial.

- $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Irrespective of the auto pad/CRC strip, the CRC module receives the entire frame to compute the CRC check for received frame.

### Receive Frame Controller Module (RFC)

The main functions of the Frame Controller are:

- Converting the 8-bit stream data to 32-bit data.
- Frame filtering.
- Attaching the calculated IP Checksum.
- Update the Receive Status.

If the RA bit of the EMAC Frame Filter register is set, the RFC module initiates the data transfer as soon as possible. At the end of the data transfer, the Frame Controller sends out the received frame status that includes the Address Filtering Pass / Fail status.

If the RA bit is reset, the Frame Controller performs frame filtering based on the destination/source address (the application still needs to perform another level of filtering if it decides not to receive any bad frames like runt, CRC error frames, etc. After receiving the destination/source address bytes, the Frame Controller checks the filter-fail signal from the AFM module for an address match. On detecting a filter-fail from AFB, the frame is dropped and not transferred to the application.

### Receive Flow Control Module (FRX)

The Receive Flow Controller detects the receiving Pause frame and pauses the frame transmission for the delay specified within the received Pause frame. The Flow Controller is enabled only in Full-Duplex mode. The Pause frame detection function can be enabled or disabled with the RFE bit of EMAC Flow Control Register.

Once the receive flow control is enabled, the Flow Controller begins monitoring the received frame destination address for any match with the multicast address of the control frame (0x0180C2000001). If a match is detected, it indicates to the Frame Controller, that the destination address of the received frame matches the reserved control frame destination address. The RFC module then decides whether or not to transfer the received control frame to the application, based on the (PCF) bit setting of EMAC Frame Filter register.

The Receive Flow Controller also decodes the Type, Op-code, and Pause Timer field of the receiving control frame. If the byte count of the frame status indicates 64 bytes, and if there is no CRC error, the Flow Controller requests the MAC transmitter to pause the transmission of any data frame for the duration of the decoded Pause Time value, multiplied by the slot time (64 byte times). Meanwhile, if another Pause frame is detected with a zero Pause Time value, the module resets the Pause Time and gives another pause



request to the Transmitter. If the received control frame matches neither the Type field (0x8808), Opcode (0x00001), nor byte length (64 bytes), or if there is a CRC error, the module does not generate a Pause request to Transmitter.

In the case of a pause frame with a multicast destination address, the Frame Controller filters the frame based on the address match from the Flow Controller. For a pause frame with a unicast destination address, the filtering in the FRX module depends on whether the Destination Address matched the contents of the MAC Address Register 0 and the UP Bit of EMAC Flow Control Register is set (detecting a pause frame even with a unicast destination address). The PCF register bits (Bit [7:6] of EMAC Frame Filter Register) controls the filtering for control frames in addition to the Address filter module.

### Receive Checksum Offload Engine (RCOE)

When checksum offloading is enabled, both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed for data integrity. User can enable this module by setting the IPC bit in the EMAC Configuration register. The EMAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification applies to VLAN-tagged frames as well. *Extended descriptor mode (8 x32-bit words) must be enabled to get the IPC checksum engine status in RDES4.* Status can be checked by polling the bit 0 of RDES0 word of Receive Descriptor and then if this bit is set, further parsing bits [7:0] of RDES4 word.

The Receive Checksum Offload engine calculates IPv4 header checksums and checks if they match the received IPv4 header checksums. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the IPv4 header's Length field (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP/UDP/ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not tally to the expected payload length given in the IP header.

**NOTE:** COE engine bypasses the payload of fragmented IP datagrams, IP datagrams with security features, IPv6 routing headers, and payloads other than TCP, UDP or ICMP. This information (whether the checksum engine is bypassed or not) is given in the receive status.

The right meaning of Checksum related errors can be understood with the below bit combination in receive descriptors (Frame Status with Full Checksum Offload Engine Enabled and Advanced Timestamps not Enabled):

**Table 23-25:** Checksum error status

| IEEE802.3 Frame: bit 5 of RDES0 | Header Checksum Error: bit 3 of RDES4 | Payload Checksum Error: bit 4 of RDES4 | Frame Status                                                                                                   |
|---------------------------------|---------------------------------------|----------------------------------------|----------------------------------------------------------------------------------------------------------------|
| 0                               | 0                                     | 0                                      | The frame is an IEEE 802.3 frame (Length field value is less than 0x0600).                                     |
| 1                               | 0                                     | 0                                      | IPv4/IPv6 Type frame in which no checksum error is detected.                                                   |
| 1                               | 0                                     | 1                                      | IPv4/IPv6 Type frame in which a payload checksum error (as described for PCE) is detected                      |
| 1                               | 1                                     | 0                                      | IPv4/IPv6 Type frame in which IP header checksum error (as described for IPC HCE) is detected.                 |
| 1                               | 1                                     | 1                                      | IPv4/IPv6 Type frame in which both PCE and IPC HCE is detected.                                                |
| 0                               | 0                                     | 1                                      | IPv4/IPv6 Type frame in which there is no IP HCE and the payload check is bypassed due to unsupported payload. |
| 0                               | 1                                     | 1                                      | Type frame which is neither IPv4 or IPv6 (COE bypasses the checksum check completely)                          |
| 0                               | 1                                     | 0                                      | Reserved                                                                                                       |

### Receive Bus Interface Unit Module (RBU)

The Receive Bus Interface Unit (RBU) constructs the 32-bit data received from the Frame Controller into a 32-bit FIFO based protocol.

### Address Filtering Module (AFM)

The Address Filtering (AFM) module performs the destination checking function on all received frames and reports the address filtering status to the Frame Controller. The address checking is based on different parameters (Frame Filter register) chosen by the application. These parameters are inputs to the AFM module as control signals, and the AFM module reports the status of the address filtering based on the combination of these inputs. The AFM module also reports whether the receiving frame is a multicast

frame or a broadcast frame, as well as the address filter status. The AFM module uses the station's physical (MAC) address and the Multicast Hash table for address checking.

- Hash or Perfect Address Filter:
  - The Destination Address filter can be configured to pass a frame when its Destination Address matches either the Hash filter or the Perfect filter by setting the HPF bit of the Frame Filter register and setting the corresponding HUC or HMC bits. This configuration applies to both unicast and multicast frames. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to the received frame.
  - *Hash filtering is not perfect filtering because a 48-bit MAC address is reduced to a 6-bit hash value. Consequently, there may be instances where more than one address has the same hash value.*
- Unicast Destination Address Filter:
  - The AFM supports 1 MAC address for unicast perfect filtering. If perfect filtering is selected (HUC bit of Frame Filter register is reset), the AFM compares all 48 bits of the received unicast address with the programmed MAC address for any match.
  - In Hash filtering mode (When HUC bit is set), the AFM performs imperfect filtering for unicast addresses using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of '000000' selects Bit 0 of the selected register, and a value of 111111 selects Bit 63 of the Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.
- Multicast Destination Address Filter:
  - The EMAC can be programmed to pass all multicast frames by setting the PM bit in the Frame Filter register. If the PM bit is reset, the AFM performs the filtering for multicast addresses based on the HMC bit of Frame Filter register. In Perfect Filtering mode, the multicast address is compared with the programmed MAC Destination Address register. Group address filtering is also supported.
  - In Hash filtering mode, the AFM performs imperfect filtering using a 64-bit Hash table. For hash filtering, the AFM uses the upper 6 bits CRC of the received multicast address to index the content of the Hash table. A value of '000000' selects Bit 0 of the selected register and a value of '111111' selects Bit 63 of the Hash Table register. If the corresponding bit is set to 1, then the multicast frame is said to have passed the Hash filter; otherwise, the frame has failed the Hash filter.
- Broadcast Address Filter:
  - The AFM doesn't filter any broadcast frames in the default mode. However, if the EMAC is programmed to reject all broadcast frames by setting the DBF bit in the Frame Filter register, the AFM asserts the Filter fail signal, whenever a broadcast frame is received.
- Inverse Filtering Operation:

- There is an option to invert the filter-match result at the final output. This is controlled by the **DAIF** bit of the Frame Filter register. The **DAIF** bit is applicable for both Unicast and Multicast DA frames. The result of the unicast / multicast destination address filter is inverted in this mode.

## Destination Address Filtering

The following table provides possible address filtering scenarios with the EMAC AFM module.

**Table 23-26:** Destination Address Filtering

| Frame Type | PR | HPF | HUC | HMC | DAIF | PM | DB | DA Filter Operation                         |
|------------|----|-----|-----|-----|------|----|----|---------------------------------------------|
| Broadcast  | 1  | X   | X   | X   | X    | X  | X  | Pass                                        |
|            | 0  | X   | X   | X   | X    | X  | 0  | Pass                                        |
|            | 0  | X   | X   | X   | X    | X  | 1  | Fail                                        |
| Unicast    | 1  | X   | X   | X   | X    | X  | X  | Pass all frames.                            |
|            | 0  | X   | 0   | 0   | X    | X  | X  | Pass on Perfect/Group filter match.         |
|            | 0  | X   | 0   | 1   | X    | X  | X  | Fail on Perfect/Group filter match.         |
|            | 0  | 0   | 1   | 0   | X    | X  | X  | Pass on Hash filter match.                  |
|            | 0  | 0   | 1   | 1   | X    | X  | X  | Fail on Hash filter match.                  |
|            | 0  | 1   | 1   | 0   | X    | X  | X  | Pass on Hash or Perfect/Group filter match. |
|            | 0  | 1   | 1   | 1   | X    | X  | X  | Fail on Hash or Perfect/Group filter match. |
| Multicast  | 1  | X   | X   | X   | X    | X  | X  | Pass all frames.                            |
|            | X  | X   | X   | X   | X    | 1  | X  | Pass all frames.                            |

**Table 23-26:** Destination Address Filtering (Continued)

| Frame Type | PR | HPF | HUC | HMC | DAIF | PM | DB | DA Filter Operation                                                                   |
|------------|----|-----|-----|-----|------|----|----|---------------------------------------------------------------------------------------|
|            | 0  | X   | X   | 0   | 0    | 0  | X  | Pass on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.         |
|            | 0  | 0   | X   | 0   | 1    | 0  | X  | Pass on Hash filter match and drop PAUSE control frames if PCF = 0x.                  |
|            | 0  | 1   | X   | 0   | 1    | 0  | X  | Pass on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x. |
|            | 0  | X   | X   | 1   | 0    | 0  | X  | Fail on Perfect/Group filter match and drop PAUSE control frames if PCF = 0x.         |
|            | 0  | 0   | X   | 1   | 1    | 0  | X  | Fail on Hash filter match and drop PAUSE control frames if PCF = 0x.                  |
|            | 0  | 1   | X   | 1   | 1    | 0  | X  | Fail on Hash or Perfect/Group filter match and drop PAUSE control frames if PCF = 0x. |

## EMAC Station Management Interface (SMI)

The IEEE 802.3 MII Station Management interface (applicable for RMII as well), also known as MDIO management interface, allows the Blackfin processor to monitor and control one or more external Ethernet physical-layer transceivers (commonly called PHYs). The management interface physically consists of a 2-wire serial connection composed of the MDC (management data clock) output signal and the MDIO (management data input/output) bidirectional data signal.

The application can select one of the 32 PHYs and one of the 32 registers within any PHY and send control data or receive status information. Only one register in one PHY can be addressed at any given time. All the transfers are initiated by the EMAC CORE, and the PHY chip only acts as a slave device.

Standard PHY control and status registers typically provide device capability status bits (for example, auto-negotiation, duplex modes, 10/100 speeds and protocols), device status bits (for example, auto-negotiation complete, link status, remote fault), and device control bits (for example, reset, speed selection, loopback, and auto-negotiation start). The features supported by the PHY may be determined at power-up by an MDIO read access (at default rates) of device capabilities in PHY status registers.

The MII management logical interface specifies:

- A set of 16-bit device control/status registers within PHYs, including both required registers with standardized bit definitions as well as optional vendor-specified registers.
- A 5-bit device addressing scheme which allows the MAC to select one of up to 32 externally-connected PHY devices.
- A 5-bit register addressing scheme for selecting the target register within the addressed device.
- A transfer frame protocol for 16-bit read and write accesses to PHY registers via the MDC and MDIO signals under control of the MAC.

**Table 23-27:** Station Management Interface pins

| Station Management Interface pins | Pin Description                                                                                                |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------|
| MDIO - Management Data IO         | A periodic clock that runs at a maximum period of 400ns. Always Driven by the EMAC to PHY.                     |
| MDC - Management Data Clock       | Data signal driven by EMAC or PHY, depending on write or read access with respect to EMAC; synchronous to MDC. |

## MDC Clock Frequency

The frequency of MDC is determined by the `CSR` field in the `SMI_ADDR` register. The clock range selection determines the frequency of the clock relative to the SCLK frequency. The suggested range of SCLK frequency applicable for each value of `CSR` field is given below. The programmability based on SCLK frequency range ensures that the MDC clock frequency range is within the IEEE specifications of 1.0 MHz - 2.4 MHz. However, EMAC MDC can also support higher frequencies for PHY devices that support the same.

**Table 23-28:** MDC clock frequency selection

| CSR Selection | Programmed SCLK frequency range | Frequency of MDC | Min and Max MDC Freq, as per specifications |
|---------------|---------------------------------|------------------|---------------------------------------------|
| 0000          | 60-100 MHz                      | SCLK/42          | MIN= 1.43MHz and MAX= 2.39 MHz              |
| 0001          | 100-125 MHz                     | SCLK/62          | MIN = 1.61MHz and MAX=2.01MHz               |
| 0010          | 20-35 MHz                       | SCLK/16          | MIN = 1.25MHz and MAX=2.19MHz               |
| 0011          | 35-60 MHz                       | SCLK/26          | MIN = 1.35MHz and MAX=2.31MHz               |

MDIO data transfer parameters are given below. Write and Read sequences provided further are based on these parameters.

**Table 23-29:** MDIO frame parameters

| Parameter | Description                                                                            |
|-----------|----------------------------------------------------------------------------------------|
| IDLE      | The MDIO line is three-state (Noted as ' Z' in sequence); there is no clock on MDC.    |
| PREAMBLE  | 32 continuous bits, each of value 1.                                                   |
| START     | Start-of-frame is 01 .                                                                 |
| OPCODE    | 10 for Read and 01 for Write.                                                          |
| PHY ADDR  | 5-bit address select for one of 32 PHYs (Noted as ' AAAAA' in sequence).               |
| REG ADDR  | Register address in the selected PHY (Noted as ' RRRRR' in sequence).                  |
| TA        | Turnaround is Z0 for Read and 10 for Write (Z = High Impedance).                       |
| DATA      | Any 16-bit value. Driven by MAC or PHY based on direction (Noted as ' DDD . . . DDD'). |

Table 23-30: MDIO Write data sequence

| IDLE | PREAMBLE  | START | OPCODE | PHY ADDR | REG ADDR | TA | DATA          | IDLE |
|------|-----------|-------|--------|----------|----------|----|---------------|------|
| Z    | 111...111 | 01    | 01     | AAAAA    | RRRRR    | 10 | DDD...D<br>DD | Z    |

Table 23-31: MDIO Read data sequence

| IDLE | PREAMBLE  | START | OPCODE | PHY ADDR | REG ADDR | TA | DATA          | IDLE |
|------|-----------|-------|--------|----------|----------|----|---------------|------|
| Z    | 111...111 | 01    | 10     | AAAAA    | RRRRR    | Z0 | DDD...D<br>DD | Z    |

SMI Write Operation

When the user sets the Write and Busy bits in SMI\_ADDR Register, the Station Management Interface initiates a Write operation into the PHY registers with the Management Frame Format (the PHY address, the register address in PHY, and the write data) specified in the IEEE specifications (Section 22.2.4.5 of IEEE Standard). The application should not change the SMI Address register contents or the SMI Data register while the transaction is ongoing. Write operations to the SMI Address register or the SMI Data Register during the transfer period are ignored (while the Busy bit is high), and the transaction is completed without any error. After the Write operation has completed, the SMI indicates the same by resetting the Busy bit. The EMAC drives the MDIO line for the complete duration of the frame.

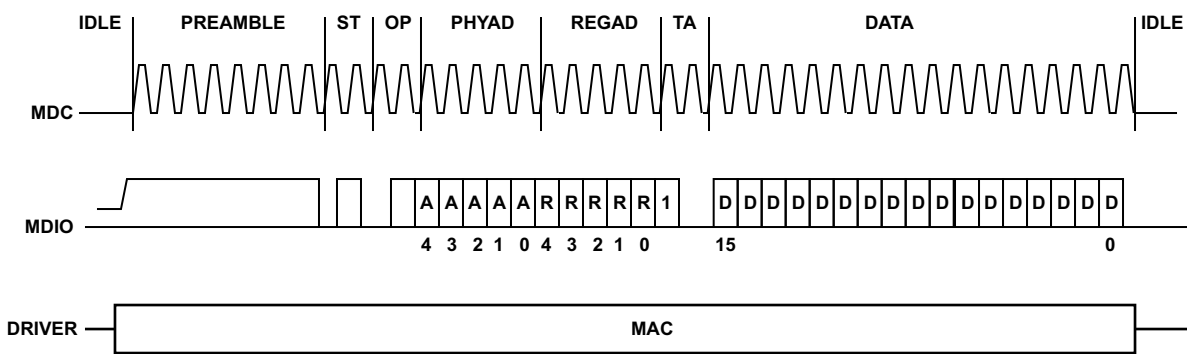


Figure 23-9: SMI Write operation via MDIO/MDC pins

SMI Read Operation

When the user sets the Busy bit in SMI\_ADDR Register with the SMI Write bit as '0', the Station Management Interface transfers the PHY address and the register address in PHY to the SMI to initiate a Read operation in the PHY registers. The application should not change the SMI Address register contents or the SMI Data register while the transaction is ongoing. Write operations to the SMI Address register or



SMI Data Register during the transfer period are ignored (while Busy bit is high) and the transaction is completed without any error. After the Read operation has completed, the SMI indicates this by resetting the Busy bit and updates the SMI Data register with the data read from the PHY. The EMAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line.

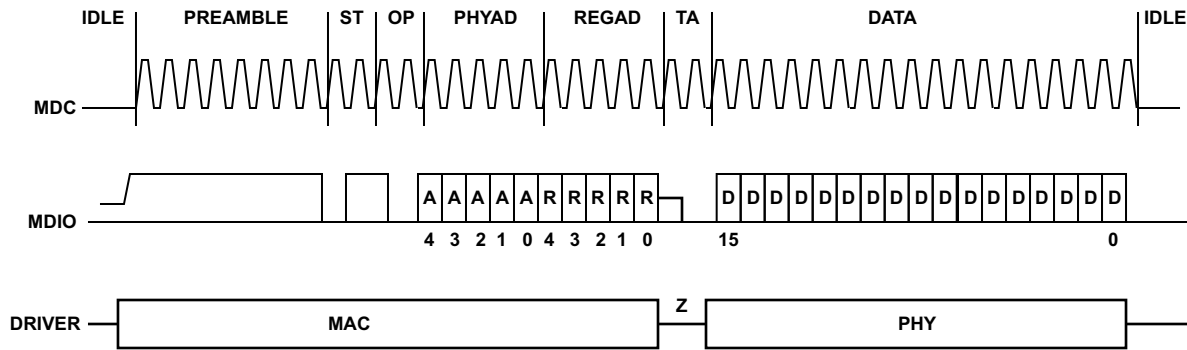


**Figure 23-10:** SMI Read operation via MDIO/MDC pins

## EMAC Management Counters (MMC)

The EMAC provides a comprehensive set of 32-bit MAC management counters. These counters are used for gathering statistics on the received and transmitted frames. The MMC sub-block also includes a control register for controlling the behavior of the counters, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt Register (receive and transmit).

The Receive MMC counters are updated for frames that are passed by the Address Filtering sub-block in EMAC CORE. Statistics of frames that are dropped by the AFM module are not updated unless they are runt frames of less than 6 bytes (Destination Address bytes are not received fully). The Module is also capable of gathering statistics on encapsulated IPv4, IPv6, and TCP, UDP, or ICMP payloads in received Ethernet frames.

Please refer to the Register Descriptions for all the statistical counters available in EMAC. The MMC register naming conventions are as follows:

- TX as a prefix or suffix indicates counters associated with transmission.
- RX as a prefix or suffix indicates counters associated with reception.
- \_G as a suffix indicates registers that count good frames only.
- \_GB as a suffix indicates registers that count frames regardless of whether they are good or bad.

Transmitted frames are considered “good” if transmitted successfully. In other words, a transmitted frame is good if the frame transmission is not aborted due to any of the following errors:

- Jabber Timeout
- No Carrier/Loss of Carrier

- Late Collision
- Frame Underflow
- Excessive Deferral
- Excessive Collision

Received frames are considered “good” if none of the following errors exists:

- CRC error
- Runt Frame (shorter than 64 bytes)
- Alignment error
- Length error (non-Type frames only)
- Out of Range (non-Type frames only, longer than maximum size)

The maximum frame size depends on the frame type, as follows:

- Untagged frame maxsize = 1518
- VLAN Frame maxsize = 1522
- Jumbo Frame maxsize = 9018
- Jumbo VLAN Frame maxsize = 9022

The Module includes a Control Register for Preset, Freeze and Roll-Over of counters. Additional configuration include `RDRST` bit for enabling an auto-reset feature whenever the Counters are read and `RST` bit for resetting all the counters. Please refer to the Register Descriptions for detailed bit level description of MMC Control Register.

The MMC can also trigger an interrupt, when the corresponding bits are enabled in the Mask Registers of Transmit, Receive and IPC, and when the particular counter reaches half/full. The status is also updated in the corresponding Interrupt register.

### **MMC Receive Interrupt Register**

The MMC Receive Interrupt Register maintains the interrupts that are generated when receive statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt Register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits [7:0]) of the respective counter must be read in order to clear the interrupt bit.

### **MMC Transmit Interrupt Register**

The MMC Transmit Interrupt Register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values

(0xFFFFFFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Transmit Interrupt Register is a 32-bit wide register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (bits [7:0]) of the respective counter must be read in order to clear the interrupt bit.

### MMC Receive Checksum Offload Interrupt Register

The MMC Receive Checksum Offload Interrupt Register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x80000000), and when they cross their maximum values (0xFFFFFFFF). When Counter Stop Rollover is set, then interrupts are set but the counter remains at all-ones. The MMC Receive Checksum Offload Interrupt register is 32-bits wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (bits [7:0]) must be read to clear the interrupt bit.

## EMAC Precision Time Protocol (PTP) Engine

The following sections describe the Precision Time Protocol engine.

### Introduction to IEEE1588 and PTP Engine

Ethernet MAC peripheral includes a PTP Engine to assist applications requiring time synchronization. PTP module is tightly integrated with EMAC CORE, in order to aid hardware time stamping defined as per IEEE1588 2002/2008 standards. Applications can make use of accurate hardware time stamps via TCP/IP stacks (if using Network layer communication) or via Ethernet device drivers (if using MAC layer communication), to exchange time information across devices connected over network.

### Basic Concept of IEEE1588

Many systems require two independent devices to operate in a time synchronized fashion. If each system were to rely solely on its own oscillator, differences between the specific characteristics and operating conditions of the individual oscillators would limit the ability of the clocks to operate synchronously. To serve applications requiring synchronized clocks, a periodic correction mechanism need to be applied.

A simple way to synchronize multiple systems is to choose one system as a Master, who will be having the best clock. Master will broadcast the clock and timing information to other systems (say slaves) and then the slaves will adjust their clock and timing according to that of Master. But, this straight forward method has limitations, such as, Master cannot broadcast the time at infinitesimal intervals, Path delay (propagation delay) exists between Master and a slave, and the delay varies between each slave and Master.

IEEE 1588 (also known as Precision Time Protocol or PTP) standard specifies a protocol used to synchronize time and clock of multiple devices, dispersed but interconnected by any communication, for example, Ethernet (IEEE 802.3). According to the protocol timing messages are exchanged between two devices (both the devices should have same representation of its system time), and then one of the device calculates its drift from other device and corrects its system time. The protocol takes care of path delay between devices and can also help synchronize clocks of multiple devices. Hence all the limitations mentioned above are not seen in this protocol.

IEEE 1588 was published in 2002 where four types of timing messages were defined: Sync, Follow\_Up, Delay\_Req, and Delay\_Resp. Here the protocol synchronizes two or more devices where one is Master and others are slaves. Sync, Follow\_Up, and Delay\_Resp messages are sent from the Master device to Slave device in the system, while Delay\_Req message is sent from a Slave device to Master device. More information on IEEE 1588-2002 is given below.

In 2008 a newer version of IEEE 1588 was introduced which provides further mechanisms to measure the peer-to-peer delay. Three additional timing messages (PdelayReq, PdelayResp, and PdelayRespFollowup) were added to implement peer-to-peer synchronization. More information on IEEE 1588-2008 is given below.

## PTP Engine in EMAC

In IEEE 1588 protocol, for calculation of drift in time between two Ethernet devices, the device should note down its system time whenever a timing message is sent or received. Due to the indeterministic delay of a node's software system, the software is unable to capture an accurate time when the message is sent or received. However, the hardware is capable of monitoring the signal on the communication media and therefore getting accurate message arrival/departure time.

The PTP (Precision Time Protocol) module, closely integrated with the EMAC module provides hardware assistance to implement both the IEEE 1588-2002 and IEEE 1588-2008 standard on Ethernet (IEEE 802.3). It takes one input clock signal as its PTP clock and maintains the timing information (called System Time) at the level of nanoseconds.

PTP module includes hardware for clock and System Time adjustment. The System Time is physically represented by Pulse -Per-Second (PPS) signal. PPS can be programmed to a fixed frequency or provide flexibility to the signal in terms of pulse width, interval, start and stop time of the signal. PTP module can be programmed to trigger an alarm interrupt when System Time reaches specified time.

PTP module can be programmed to detect different types of received frames, capture System Time and timestamp those frames with captured System Time. User can program any frame such that PTP module will capture the System Time when it is transmitted. PTP module can also capture System Time when an event is detected on Auxiliary Snapshot Trigger input pin.

## IEEE 1588-2002 Overview

The IEEE 1588-2002 standard defines a protocol, Precision Time Protocol (PTP) that enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into Master and Slave nodes for distributing the timing/clock information. Figure below shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

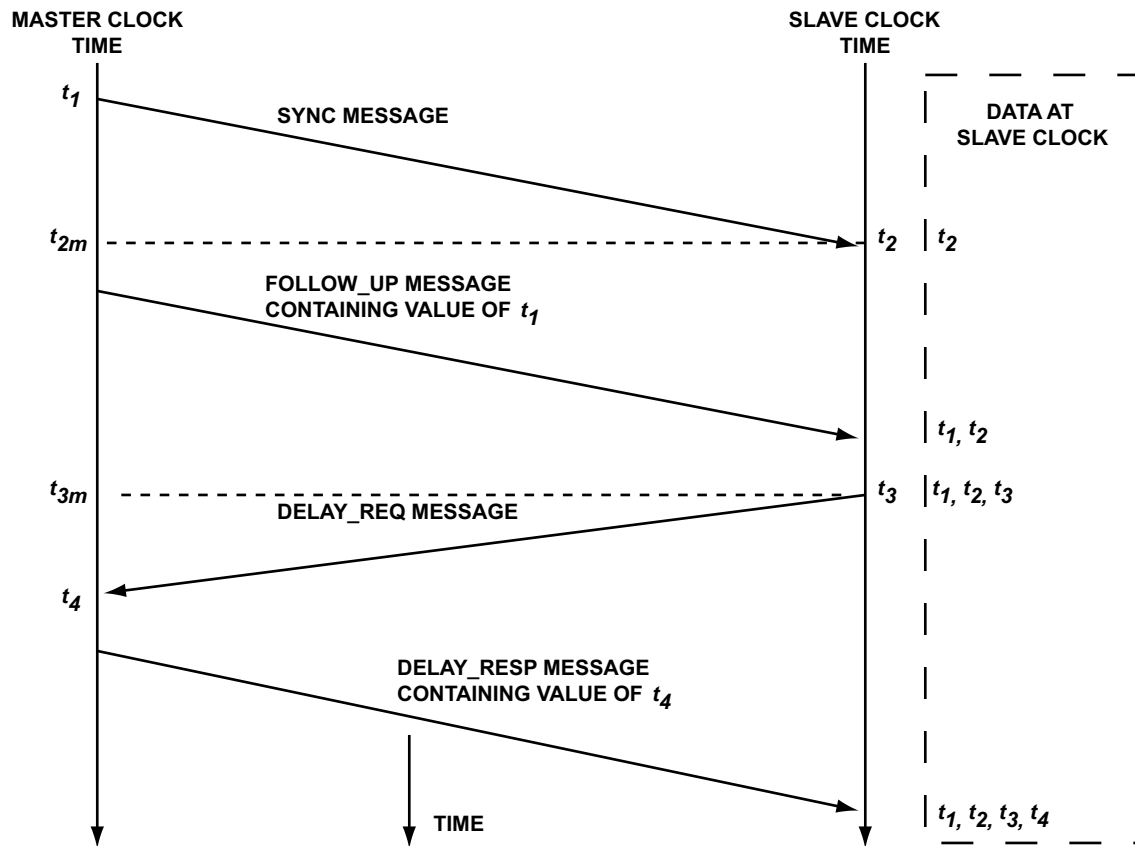


**Figure 23-11:** IEEE 1588-2002 PTP Process

As shown in Figure above, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is  $t_1$ . This time must be captured by the Master, for Ethernet ports, at RMII.
2. The slave receives the Sync message and also captures the exact time,  $t_2$ , using its timing reference.
3. The master sends a Follow\_up message to the slave, which contains  $t_1$  information for later use.
4. The slave sends a Delay\_Req message to the master, noting the exact time,  $t_3$ , at which this frame leaves the RMII.
5. The master receives the message, capturing the exact time,  $t_4$ , at which it enters its system.
6. The master sends the  $t_4$  information to the slave in the Delay\_Resp message.
7. The slave uses the four values of  $t_1$ ,  $t_2$ ,  $t_3$ , and  $t_4$  to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at

the RMI. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.

### **IEEE 1588-2008 Advanced Timestamps**

In addition to the basic timestamp features mentioned in IEEE 1588-2002 Timestamps, the EMAC supports the following advanced timestamp features defined in the IEEE 1588-2008 standard.

- Support for the IEEE 1588-2008 (Version 2) timestamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to select the node to be a Master or Slave.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to run nanoseconds time in digital or binary format.

### **Peer-to-Peer (P2P) PTP Message Support**

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. Figure below shows the method to calculate the propagation delay between nodes supporting peer-to-peer path correction.

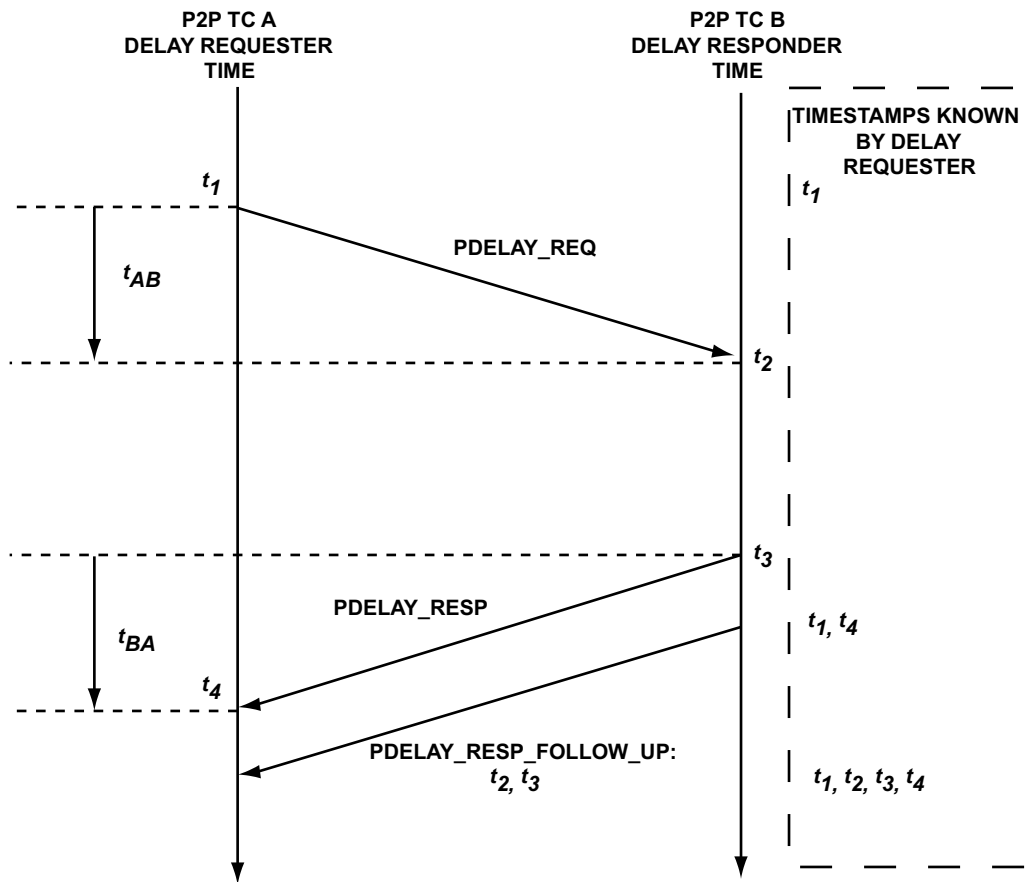


**Figure 23-12:** Propagation Delay Calculation between Nodes Supporting Peer-to-Peer Path Correction

As shown in Figure above, the propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay\_Req message and generates a timestamp,  $t_1$ , for the Pdelay\_Req message.
2. Port-2 receives the Pdelay\_Req message and generates a timestamp,  $t_2$ , for this message.
3. Port-2 returns a Pdelay\_Resp message and generates a timestamp,  $t_3$ , for this message. To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay\_Resp message as quickly as possible after the receipt of the Pdelay\_Req message. The Port-2 returns any one of the following:
  - The difference between the timestamps  $t_2$  and  $t_3$  in the Pdelay\_Resp message.
  - The difference between the timestamps  $t_2$  and  $t_3$  in the Pdelay\_Resp\_Follow\_Up message.
  - The timestamps  $t_2$  and  $t_3$  in the Pdelay\_Resp and Pdelay\_Resp\_Follow\_Up messages respectively.
4. Port-1 generates a timestamp,  $t_4$ , on receiving the Pdelay\_Resp message.

Port-1 uses all four timestamps to compute the mean link delay.

## Block Diagram

The following figure shows the functional Block diagram of PTP module.

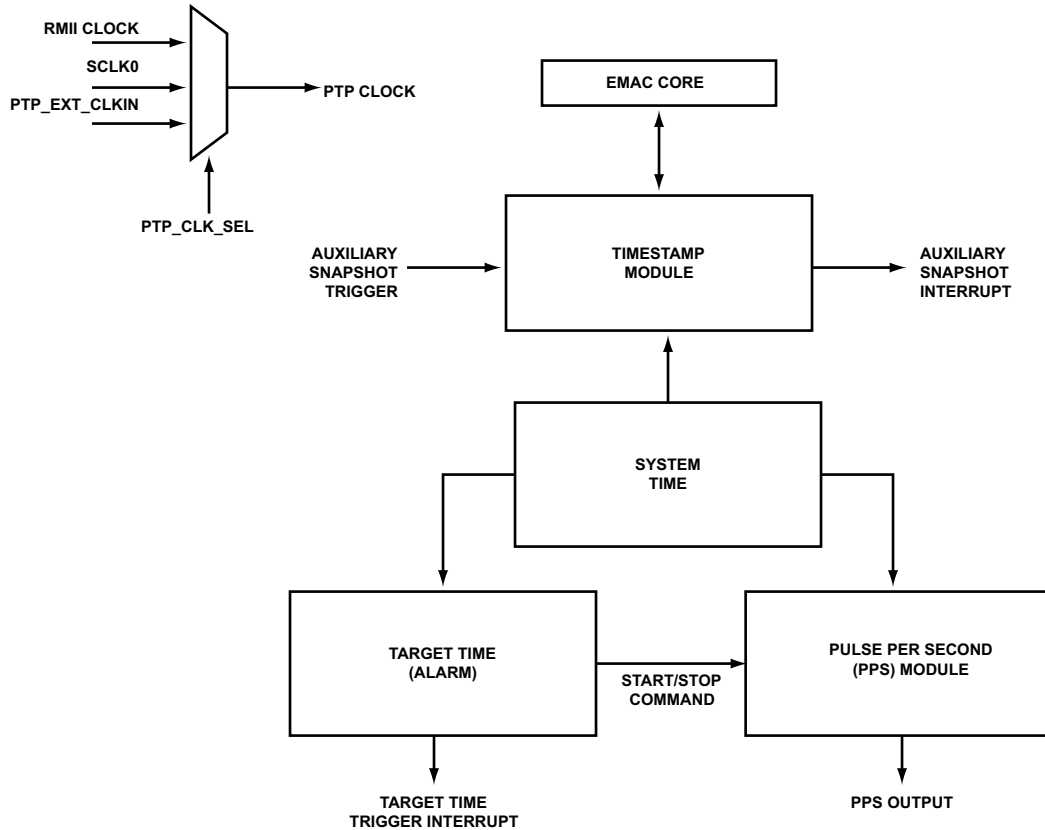


**Figure 23-13:** PTP Block Diagram

A System Time module is present which keeps the time of PTP module. It consists of hardware which can be programmed by the user for time initialization, time correction and clock correction.

The Timestamp module is capable of capturing the time (provided by System Time module) at various conditions such as, a frame is sent or received by EMAC, rising edge at Auxiliary Snapshot Trigger pin. When System Time is captured after detection of a frame, Timestamp module automatically includes the time information in the frame descriptor. Timestamp on detection on frame can be programmed on per frame basis. Timestamp module also captures the System Time when a rising edge occurs on Auxiliary Snapshot Trigger pin.

PTP module is driven by PTP clock. This clock can be chosen between 3 different clock sources.

Pulse per Second (PPS) module is used to generate a pulse or train of pulse on PPS Output pin, and it is the physical representation of System Time. PPS can be fixed (where only frequency can be varied) or flexible (where width, interval, start time and stop time can be programmed).



Target Time module acts as an Alarm for PTP module. Whenever System Time reaches value equal to programmed Target Time, Target Time trigger interrupt is generated. By appropriate programming, Target Time trigger can also be used to Start or Stop flexible PPS output at specific time.

## PTP Module Clock

The PTP module clock features include *PTP Clock Source Selection* and *PTP Clock Frequency Range*.

### PTP Clock Source Selection

The PTP module can take one of three clock sources as its input clock — SCLK0, RMII Clock or PTP external clock.

PTP clock source can be selected using EMACx field in PADS0\_EMAC\_PTP\_CLKSEL register, as shown in Table below.

| EMACx field | PTP clock source   | Clock Description                  |
|-------------|--------------------|------------------------------------|
| 00          | EMACx_RMII         | RMII reference clock               |
| 10          | PTP External Clock | Clock available on PTP_EXT_CLK pin |
| X1          | SCLK0              | Processor System Clock 0           |

### PTP Clock Frequency Range

The resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance. The maximum PTP clock frequency is limited by the timing constraints achievable for logic operating on the PTP clock. The maximum PTP clock frequency for different PTP clock source is as given below.

| PTP clock source | Maximum PTP clock frequency (MHz) |
|------------------|-----------------------------------|
| SCLK0            | 125                               |
| RMII             | 50                                |
| PTP_EXTCLK       | 83                                |

The minimum PTP clock frequency depends on the time required between two consecutive frames. Because the RMII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC as shown in Table below.

A minimum delay required between two consecutive timestamp captures is 8 clock cycles of RMII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the EMAC does not take a timestamp snapshot for the second frame.

| Mode                           | Minimum Gap Between Two frames                                                                     | Minimum PTP Frequency                                                                                                                                                                                 |
|--------------------------------|----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100-Mbps full-duplex operation | 336 RMII clocks<br>(256 clocks for a 64-byte frame + 48 clocks of min IFG + 32 clocks of preamble) | $(3 * \text{PTP period}) + (8 * \text{RMII period}) \leq (336 * \text{RMII period})$<br>Maximum PTP period = $(336 - 8) * 20 \text{ ns} \div 3 = 2,186 \text{ ns}$<br>Minimum PTP frequency = 0.46MHz |

## Timestamp Module

Timestamp module captures time in seconds and nanoseconds maintained as System Time. Timestamp module captures time when specific events occur. Events include detection of a frame transmitted or received over EMAC and rising edge on Auxiliary Snapshot Trigger pin. Timestamp module need not timestamp all the frames sent or received over EMAC. PTP module can be programmed to detect specific kinds of frames for timestamping.

## Frame Detection and Timestamping

The PTP module automatically monitors all received and transmitted IEEE 1588 event messages on the Ethernet. If an event message is detected, the module takes a snapshot of the System Time and stores its value to the 64-bit fields in Transmit or Receive Descriptor. The timestamping is done at the EMAC RMII interface when the module sees the start of frame of an event message packet.

## Transmit Path Timestamping

EMAC captures a timestamp when a frame is being sent on RMII. Timestamp capture is controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame or not.

Application should extend the descriptor word length from 4 WORDS to 8 WORDS by setting ATDS (bit7) in EMACx\_DMA\_BUSMODE register. In order to enable Timestamping functionality TTSE (Transmit Timestamp Enable) bit in Transmit Descriptor Word TDES0 should be set. When PTP module captures a Timestamp of a transmitted frame, it is notified to the application by setting the TTSS (Transmit Timestamp Status) in TDES0. The EMAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific frame. The 64-bit timestamp information is written to the TDES6 and TDES7 fields. The TDES6 field holds the 32 least significant bits of the timestamp (System Time Nanoseconds), except as described in Transmit Timestamp Field and TDES7 field holds 32 most significant bits (System Time Seconds). After PTP module timestamps the frame, the application can get the timestamp along with the Transmit status from EMAC.

Note: PTP module will timestamp all the transmitting frames that has TTSE set in its TDES0. It will not distinguish according to type of transmitting frame.

## Receive Path Timestamping

PTP module captures the timestamp of received frames on the RMII. Timestamp capture is controllable on a per-frame and per-type basis. In other words each received frame is timestamped according to the frame type.

Applications should extend the descriptor word length from 4 WORDS to 8 WORDS by setting ATDS (bit 7) in EMACx\_DMA\_BUSMODE register, in order to store timestamp and received message status. Receive Time Stamp availability is notified to application when PTP module sets bit 7 (Timestamp Available) in Receive Descriptor Word RDES0. When bit 0 (Extended Status Available) is set in RDES0, it indicates that extended status of PTP frame is provided in RDES4 word. Extended status include PTP Version, PTP Frame Type and Message Type. EMAC returns the timestamp to the software inside the corresponding receive descriptor. The 64-bit timestamp information is written back to the RDES6 and RDES7 fields. The RDES6 holds the 32 least significant bits of the timestamp (System Time Nanoseconds), except as mentioned in Receive Timestamp Field and RDES7 field holds 32 most significant bits (System Time Seconds).

The timestamp is written only to that receive descriptor for which the Last Descriptor status field has been set to 1. When the timestamp is not available (for example, because of an RxFIFO overflow), an all-ones pattern is written to the descriptors (RDES6 and RDES7), indicating that timestamp is not correct. RDES0 [7] will indicate whether the time-stamp is updated in RDES6/7 or not.

Processing of received frames to identify valid PTP frames is done by PTP module. The snapshot of the time to be sent to the application can be controlled using EMACx\_TM\_CTL register.

The PTP module can be programmed to detect all received frames or only some types of PTP frames, according to fields present in EMACx\_TM\_CTL register. Refer to the table given below:

| TSENALL(bit 8) | SNAPTYPSEL(bits [17:16]) | TSMSTRENA(bit 15) | TSEVNTENA(bit 14) | Frames                                 |
|----------------|--------------------------|-------------------|-------------------|----------------------------------------|
| 1              | X                        | X                 | X                 | All                                    |
| 0              | 00                       | X                 | 0                 | Sync, Follow_Up, Delay_Req, Delay_Resp |
| 0              | 00                       | 0                 | 1                 | Sync                                   |
| 0              | 00                       | 1                 | 1                 | Delay_Req                              |

|   |    |   |   |                                                                                                       |
|---|----|---|---|-------------------------------------------------------------------------------------------------------|
| 0 | 01 | X | 0 | Sync, Follow_Up,<br>Delay_Req,<br>Delay_Resp,<br>Pdelay_Req,<br>Pdelay_Resp,<br>Pdelay_Resp_Follow_Up |
| 0 | 01 | 0 | 1 | Sync, Pdelay_Req,<br>Pdelay_Resp                                                                      |
| 0 | 01 | 1 | 1 | Delay_Req,<br>Pdelay_Req,<br>Pdelay_Resp                                                              |
| 0 | 10 | X | X | Sync, Delay_Req                                                                                       |
| 0 | 11 | X | X | Pdelay_Req,<br>Pdelay_Resp                                                                            |

## PTP Processing and Control

When EMAC receives a frame, Frame detection and Timestamping by Timestamp module of PTP is done on the basis of some of the PTP fields in the frame. *PTP Message Format defined in IEEE 1588-2008* shows the common message header for the PTP messages. This format is taken from IEEE standard 1588-2008. When EMAC needs to send a PTP frame, the frame has to follow this format.

When a frame is received, PTP module compares these fields with standard values and finds out the type of PTP frame and other information such as PTP version, IP version, etc. It then updates the related fields in RDES4. When a frame is transmitted user should make sure that all the fields are appropriate such that a PTP module on the other end of communication can correctly detect and decode the frame.

**Table 23-32:** PTP Message Format defined in IEEE 1588-2008

| Bits              |   |   |   |             |   |   |   | Octets | Offset |
|-------------------|---|---|---|-------------|---|---|---|--------|--------|
| 7                 | 6 | 5 | 4 | 3           | 2 | 1 | 0 |        |        |
| transportSpecific |   |   |   | messageType |   |   |   | 1      | 0      |
| Reserved          |   |   |   | versionPTP  |   |   |   | 1      | 1      |
| messageLength     |   |   |   |             |   |   |   | 2      | 2      |
| domainNumber      |   |   |   |             |   |   |   | 1      | 4      |

**Table 23-32:** PTP Message Format defined in IEEE 1588-2008 (Continued)

| Bits               | Octets | Offset |
|--------------------|--------|--------|
| Reserved           | 1      | 5      |
| flagField          | 2      | 6      |
| correctionField    | 8      | 8      |
| Reserved           | 4      | 16     |
| sourcePortIdentity | 10     | 20     |
| sequenceId         | 2      | 30     |
| controlField (*)   | 1      | 32     |
| logMessageInterva  | 1      | 33     |

(\*) – controlField is used in version 1. For version 2, messageType field will be used for detecting different message types.

There are some fields in the Ethernet payload that user can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4
- PTP Frames Over IPv6
- PTP Frames Over Ethernet

For any of the above PTP frames, EMAC does not consider the PTP version 1 messages as valid PTP messages when frame consists of Peer delay multicast address as destination address (DA).

## PTP Frame Over IPv4

*IPv4-UDP PTP Frame Fields Required for Control and Status* provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on IEEE 1588-2008 standards and the message format defined in *PTP Message Format defined in IEEE 1588-2008*.

**Table 23-33:** IPv4-UDP PTP Frame Fields Required for Control and Status

| Field Matched  | Octet Position | Matched Value | Description   |
|----------------|----------------|---------------|---------------|
| MAC Frame type | 12, 13         | 0x0800        | IPv4 datagram |

**Table 23-33:** IPv4-UDP PTP Frame Fields Required for Control and Status (Continued)

| Field Matched                              | Octet Position | Matched Value                                                | Description                                                                                                                                                                                           |
|--------------------------------------------|----------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IP Version and Header Length               | 14             | 0x45                                                         | IP version is IPv4                                                                                                                                                                                    |
| Layer 4 Protocol                           | 23             | 0x11                                                         | UDP                                                                                                                                                                                                   |
| IP Multicast Address (IEEE 1588 Version 1) | 30, 31, 32, 33 | 0xE0,0x00, 0x01,0x81 (or 0x82 or 0x83 or 0x84)               | Multicast IPv4 addresses allowed.<br>224.0.1.129<br>224.0.1.130<br>224.0.1.131<br>224.0.1.132                                                                                                         |
| IP Multicast Address (IEEE 1588 Version 2) | 30, 31, 32, 33 | 0xE0, 0x00, 0x01, 0x81 (Hex)<br>0xE0, 0x00, 0x00, 0x6B (Hex) | PTP-Primary multicast address: 224.0.1.129<br>PTP-Peer delay multicast address: 224.0.0.107                                                                                                           |
| UDP Destination Port                       | 36, 37         | 0x013F<br>0x0140                                             | 0x013F - PTP Event Messages (*)<br><br>0x0140 - PTP general messages                                                                                                                                  |
| PTP Control Field (IEEE version 1)         | 74             | 0x00/0x01/0x02/ 0x03/0x04                                    | 0x00 - SYNC<br>0x01 - Delay_Req<br>0x02 - Follow_Up<br>0x03 - Delay_Resp<br>0x04 - Management                                                                                                         |
| PTP Message Type Field (IEEE version 2)    | 42 (nibble)    | 0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD                     | 0x0 - SYNC<br>0x1 - Delay_Req<br>0x2 - Pdelay_Req<br>0x3 - Pdelay_Resp<br>0x8 - Follow_Up<br>0x9 - Delay_Resp<br>0xA - Pdelay_Resp_Follow_Up<br>0xB - Announce<br>0xC - Signaling<br>0xD - Management |
| PTP Version                                | 43 (nibble)    | 0x1 or 0x2                                                   | 0x1 - Supports PTP version 1<br>0x2 - Supports PTP version 2                                                                                                                                          |

(\*) - PTP event messages are SYNC, Delay\_Req (IEEE 1588 version 1 and 2) or Pdelay\_Req, Pdelay\_Resp (IEEE 1588 version 2 only)

## PTP Frame Over IPv6

*IPv6-UDP PTP Frame Fields Required for Control and Status* provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on IEEE 1588-2008 standards and the message format defined in *PTP Message Format defined in IEEE 1588-2008*.

**Table 23-34:** IPv6-UDP PTP Frame Fields Required for Control and Status

| Field Matched                              | Octet Position | Matched Value                                             | Description                                                                                                                                                                                           |
|--------------------------------------------|----------------|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAC Frame type                             | 12, 13         | 0x86DD                                                    | IP datagram                                                                                                                                                                                           |
| IP Version                                 | 14(bits [7:4]) | 0x06                                                      | IP version is IPv6                                                                                                                                                                                    |
| Layer 4 Protocol                           | 20 (*)         | 0x11                                                      | UDP                                                                                                                                                                                                   |
| PTP Multicast Address                      | 38–53          | FF0x:0:0:0:0:0:0:181 (Hex)<br>FF02:0:0:0:0:0:0:0:6B (Hex) | PTP - primary multicast address:<br>FF0x:0:0:0:0:0:0:0:181 (Hex)<br>PTP - Peer delay multicast<br>address: FF02:0:0:0:0:0:0:0:6B<br>(Hex)                                                             |
| UDP Destination Port                       | 56, 57 (*)     | 0x013F, 0x0140                                            | 0x013F - PTP event messages<br><br>0x0140 - PTP general messages                                                                                                                                      |
| PTP Control Field<br>(IEEE 1588 version 1) | 93 (*)         | 0x00/0x01/0x02/ 0x03/0x04                                 | 0x00 - SYNC<br>0x01 - Delay_Req<br>0x02 - Follow_Up<br>0x03 - Delay_Resp<br>0x04 - Management (version1)                                                                                              |
| PTP Message Type<br>Field (IEEE version 2) | 74(*) (nibble) | 0x0/0x1/0x2/0x3/0x8/0x9/0xA/0xB/ 0xC/0xD                  | 0x0 - SYNC<br>0x1 - Delay_Req<br>0x2 - Pdelay_Req<br>0x3 - Pdelay_Resp<br>0x8 - Follow_Up<br>0x9 - Delay_Resp<br>0xA - Pdelay_Resp_Follow_Up<br>0xB - Announce<br>0xC - Signaling<br>0xD - Management |

**Table 23-34:** IPv6-UDP PTP Frame Fields Required for Control and Status (Continued)

| Field Matched | Octet Position | Matched Value | Description                                                  |
|---------------|----------------|---------------|--------------------------------------------------------------|
| PTP Version   | 75(nibble)     | 0x1 or 0x2    | 0x1 - Supports PTP version 1<br>0x2 - Supports PTP version 2 |

(\*) - IPv6 Extension Header is not defined for PTP packets

## PTP Frame Over Ethernet

*Ethernet PTP Frame Fields Required for Control and Status* provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on IEEE 1588-2008 standards and the message format defined in *PTP Message Format defined in IEEE 1588-2008*.

**Table 23-35:** Ethernet PTP Frame Fields Required for Control and Status

| Field Matched                           | Octet Position | Matched value                                | Description                                                                                                                                                                                           |
|-----------------------------------------|----------------|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MAC Destination Multicast Address(*)    | 0–5            | 01-1B-19-00-00-00<br>01-80-C2-00-00-0E       | All PTP messages can use any of the following multicast addresses:<br>01-1B-19-00-00-00<br>01-80-C2-00-00-0E                                                                                          |
| MAC Frame Type                          | 12, 13         | 0x88F7                                       | PTP Ethernet frame                                                                                                                                                                                    |
| PTP control field (IEEE Version 1)      | 45             | 0x00/0x01/0x02/<br>0x03/0x04                 | 0x00 - SYNC<br>0x01 - Delay_Req<br>0x02 - Follow_Up<br>0x03 - Delay_Resp<br>0x04 - Management                                                                                                         |
| PTP Message Type Field (IEEE version 2) | 14(nibble)     | 0x0/0x1/0x2/0x3/0x8/0<br>x9/0xA/0xB/ 0xC/0xD | 0x0 - SYNC<br>0x1 - Delay_Req<br>0x2 - Pdelay_Req<br>0x3 - Pdelay_Resp<br>0x8 - Follow_Up<br>0x9 - Delay_Resp<br>0xA - Pdelay_Resp_Follow_Up<br>0xB - Announce<br>0xC - Signaling<br>0xD - Management |



**Table 23-35:** Ethernet PTP Frame Fields Required for Control and Status (Continued)

| Field Matched | Octet Position | Matched value | Description                                                  |
|---------------|----------------|---------------|--------------------------------------------------------------|
| PTP Version   | 15(nibble)     | 0x1 or 0x2    | 0x1 - Supports PTP version 1<br>0x2 - Supports PTP version 2 |

(\*) - The address match of destination address (DA) programmed in MAC address 0 is used if TSENMACADDR (bit 18) in EMACx\_TM\_CTL register is set.

## Auxiliary Timestamp Snapshot

The auxiliary snapshot feature allows storing snapshots of the system time whenever a rising edge is detected on PTP\_AUXIN pin.

PTP stores these snapshots in a 4-deep FIFO. 64-bits of captured timestamp are stored in the FIFO. When a snapshot is stored, PTP indicates this to the EMAC with an interrupt (Auxiliary Snapshot interrupt) and ATSTS (bit 2) in EMACx\_TM\_STMPSTAT register is set. The value of the snapshot is read through EMACx\_TM\_AUXSTMP\_SEC and EMACx\_TM\_AUXSTMP\_NSEC registers. If the FIFO becomes full and an external trigger to take the snapshot is asserted, then a snapshot trigger-missed status (ATSSTM (bit 24)) is set in EMACx\_TM\_STMPSTAT register. The latest snapshot is not written to the FIFO when it is full.

When a host reads the 64-bit timestamp from the FIFO through EMACx\_TM\_AUXSTMP\_SEC and EMACx\_TM\_AUXSTMP\_NSEC registers, the space becomes available to store the next snapshot.

**NOTE:** A space in the FIFO is created whenever EMACx\_TM\_AUXSTMP\_SEC register is read. Hence EMACx\_TM\_AUXSTMP\_NSEC register should be read before reading EMACx\_TM\_AUXSTMP\_SEC register.

User can clear the FIFO by setting the ATSFC (bit 24) in EMACx\_TM\_CTL register. When multiple snapshots are present in the FIFO, the count is indicated in ATSNS (bits [27:25]) of EMACx\_TM\_STMPSTAT register.

**NOTE:** The minimum gap between two events on PTP\_AUXIN must be (4 cycles of PTP\_CLK + 3 cycles of SCLK0). Otherwise, the rising-edge of the trigger is missed by the logic.

## System Time

To get a snapshot of the time, the EMAC requires a reference time in 64-bit format as defined in the IEEE 1588 specification. PTP module maintains 80-bit time, known as System Time and it is updated using the PTP clock.

The 80-bit timing reference is split into the following three registers:

- 32 bit Nanoseconds (EMACx\_TM\_NSEC) register which provides time in nanoseconds
- 32 bit Seconds (EMACx\_TM\_SEC) register which provides time in seconds
- 16 bit High Seconds (EMACx\_TM\_HISEC) register which provides time beyond Seconds register. This register is not included in IEEE 1588 standard, and its use is application specific.

64-bit System Time (Seconds and Nanoseconds) is the source for taking timestamps for Ethernet frames being transmitted or received at the RMII.

Since the PTP clock frequency does not correspond to 1ns period, Nanoseconds register should be incremented with a value equal to PTP clock period for every PTP clock. This is achieved by use of EMACx\_TM\_SUBSEC register. Nanoseconds is incremented with value programmed in EMACx\_TM\_SUBSEC register every PTP clock.

Whenever EMACx\_TM\_SEC register overflow from 0xFFFFFFFF to 0x00000000, Seconds overflow interrupt is triggered and indicated by TSSOVF (bit0) in EMACx\_TM\_STMPSTAT register. After Seconds overflow EMACx\_TM\_HISEC register increments by one.

System Time module supports two types of rollover modes for EMACx\_TM\_NSEC register, Digital rollover and Binary rollover. In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9AC9FF, that is,  $10^9$  nanoseconds. After it reaches this value EMACx\_TM\_SEC register is incremented and EMACx\_TM\_NSEC register restarts count from zero. In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFFFFFF. Accuracy in binary rollover mode is ~0.466 ns per bit, whereas in digital rollover mode it is 1ns per bit.

## System Time Adjustment

The following sections describe the process for system time adjustment.

### System Time Initialization

The System Time can be initialized with 64-bit time when the PTP module is enabled. The initial value is written to the EMACx\_TM\_SECUPDT and EMACx\_TM\_NSECUPDT registers. The System Time counter is written with the value in the System Time Update registers when TSINIT (bit 2) in EMACx\_TM\_CTL register is set.

### Coarse Correction Method

If slave System Time has an offset with respect to the master's System Time, then it can be corrected using Coarse Correction Method. The time offset value is written to the EMACx\_TM\_SECUPDT and EMACx\_TM\_NSECUPDT registers. The offset value is then added to or subtracted from the System Time when TSUPDT (bit 3) in EMACx\_TM\_CTL register is set. Addition or Subtraction can be chosen by using ADDSUB (bit 31) in EMACx\_TM\_NSECUPDT register. System time correction is done in one clock in Coarse Correction Method.

### Fine Correction Method

If slave PTP clock's frequency has a drift with respect to the master PTP clock (as defined in IEEE 1588), then it can be corrected using Fine Correction Method. Here System Time is corrected over a period of time unlike as in Coarse Correction Method where it is done in one clock. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the EMACx\_TM\_ADDEND register, as shown in figure below. The arithmetic carry that the accumulator generates is used as a pulse to increment

the System Time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency divider.

This algorithm for Fine Correction Method is depicted in figure below.

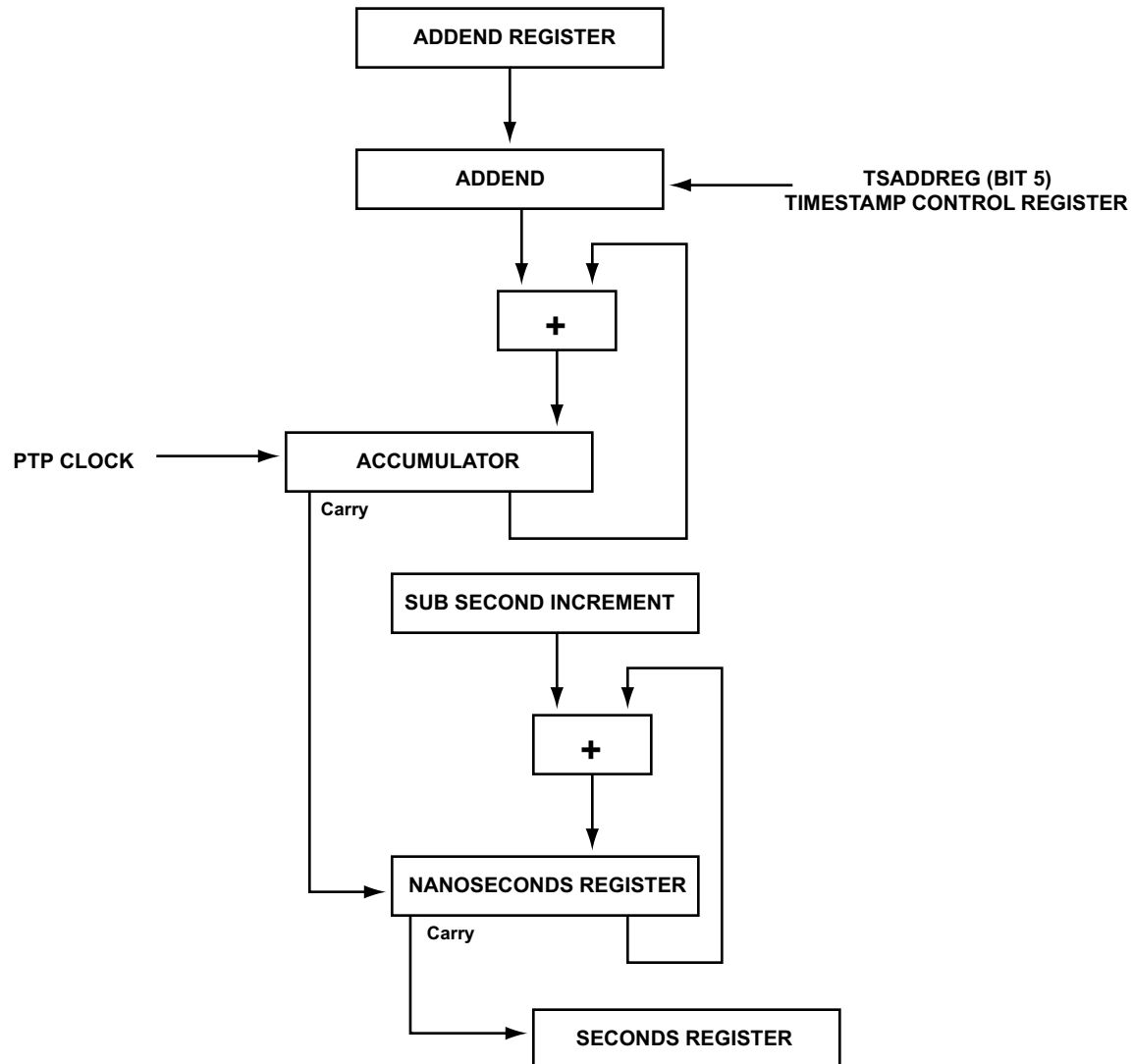


**Figure 23-14:** System Time update using Fine Correction Method

### Example for Calculation of Addend Value

Let the Master clock be 50MHz and the Slave clock be drifted to 66MHz. The aim is to run Slave System Time at 50MHz, so that Slave PTP module is synchronized with Master. If we consider the above diagram, Nanoseconds Increment signal should run at 50MHz. Nanoseconds Increment is the carry from Accumulator register, which is incremented by Addend value at rate of Slave clock (66MHz).

Say Accumulator overflows and generates carry every N Addend values. Hence,  $N * \text{Addend} = 2^{32}$ .

Accumulator is incrementing at 66MHz and carry need to be generated at 50MHz.

Hence,  $N = 66/50 = 1.32$ .

Hence  $\text{Addend} = 2^{32}/1.32 = 0xC1F07C1F$ .

Therefore, if Addend is programmed with 0xC1F07C1F, Slave System Time runs on 50MHz which is synchronized with Master.

In the figure above, the Sub Second Increment is the value which increments Nanoseconds Register according to the frequency of Nanoseconds Increment signal. Sub Second Increment is the value programmed in `EMACx_TM_SUBSEC` register. In the above example, Sub Second Increment should be 20 (for Digital Rollover) or 43 (for Binary Rollover). This will increment `EMACx_TM_NSEC` register by 20ns (1/50MHz).

The software must calculate the drift in frequency and update the `EMACx_TM_ADDEND` register accordingly.

Note: The PTP Reference clock mentioned in the following sections is the clock at which the system time gets updated. When the `TSCFUPDT` (bit 1) of `EMACx_TM_CTL` register is set to 0, this clock is same as the PTP clock. In the Fine Correction mode, PTP reference clock is the clock generated on Nanoseconds Increment signal, at which the system time gets updated.

## Target Time Trigger (Alarm)

The PTP module provides alarm functionality by triggering an alarm at a preset time. It sets `TSTARGET` (bit 1) in the `EMACx_TM_STMPSTAT` register when the System Time matches the value of the `EMACx_TM_TGTM` and `EMACx_TM_NTGTM` registers. This trigger can be used to generate an interrupt and/or command the Flexible PPS module to start/stop PPS output, depending on value programmed in `TRGTMODSEL` (bits [6:5]) in `EMACx_TM_PPSCTL` register. The trigger is enabled by setting `TSTRIG` (bit 4) in `EMACx_TM_CTL` register. Once an alarm has occurred, if another alarm is needed, the software must clear the status bit, reprogram the `EMACx_TM_TGTM` and `EMACx_TM_NTGTM` registers to a future value and set `TSTRIG` (bit 4) in `EMACx_TM_CTL` register.

If the time programmed in Target Time registers is already elapsed, then Target Time programming error is indicated by setting `TSTRGTERR` (bit 3) in `EMACx_TM_STMPSTAT` register.

The alarm time is represented in absolute units, not relative units. For example, if the software needs to generate an alarm after 10 seconds, it must read the current System Time value, add the number corresponding to 10 seconds, and write the result back to the Target Time registers.

## Pulse-Per-Second (PPS)

Pulse-per-second (PPS) is a physical representation of System Time. It is composed of a single pulse or train of pulses. PPS can be used for additional synchronization method or to monitor the synchronization performance between clocks. With proper configuration, the PTP module can be programmed to generate PPS signal which will be output on PTPxPPS pin. PTP supports two kinds of PPS Output, Fixed PPS and Flexible PPS.

## Fixed Pulse-Per-Second Output

EMAC supports fixed pulse-per-second (PPS) output that is given to indicate 1 second interval (default). The frequency of the PPS output can be changed by configuring PPSCTL (bits[3:0]) in EMACx\_TM\_PPSCTL register. The default value of PPSCTL is 0000, for which the PPS output is 1Hz signal with pulse width equals period of PTP clock.

The table below shows various frequencies achievable on PPS Output by varying PPSCTL.

| PPSCTL | Binary Rollover | Digital Rollover |
|--------|-----------------|------------------|
| 0001   | 2 Hz            | 1 Hz             |
| 0010   | 4 Hz            | 2 Hz             |
| 0011   | 8 Hz            | 4 Hz             |
| ...    | ...             | ...              |
| 1111   | 32.768 KHz      | 16.384 KHz       |

In the binary rollover mode, the PPS output has a duty cycle of 50 percent with these frequencies.

In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. PPS output pulses will have different period and duty cycle and this behavior is because of the non-linear toggling of the bits in the digital rollover mode. For example:

- When PPSCTL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.
- When PPSCTL = 0010, the PPS (2 Hz) is a sequence of: - One clock of 50 percent duty cycle and 537 ms period - Second clock of 463 ms period (268 ms low and 195 ms high).
- When PPSCTL = 0011, the PPS (4 Hz) is a sequence of: - Three clocks of 50 percent duty cycle and 268 ms period - Fourth clock of 195 ms period (134 ms low and 61 ms high).

## Flexible Pulse-Per-Second Output

EMAC also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the PPS output. This feature is called Flexible PPS and can be enabled by setting PPSSEN (bit 4) in EMACx\_TM\_PPSCTL register.

EMAC provides the following features with the Flexible PPS output option:

- Supports programming the start point of the single pulse and start and stop points of the pulse train in terms of System Time. The existing Target Time registers are used to program the start and stop time.
- Supports programming the stop time in advance, that is, user can program the stop time before the actual start time has elapsed.

- Supports programming the width, between the rising edge and corresponding falling edge of PPS signal output, in terms of number of units of sub-second increment value programmed in EMACx\_TM\_SUBSEC register.
- Supports programming the interval, between the rising edges of PPS signal, in terms of number of units of sub-second increment value programmed in EMACx\_TM\_SUBSEC register.
- Provides the option to cancel the programmed PPS start or stop request.
- Indicates error if the start or stop time being programmed has already elapsed.

### PPS Start or Stop Time

Start time can be initially programmed in the Target Time registers. If required, the start or stop time can be programmed again but it can be done only after the earlier programmed value is synchronized to the PTP clock domain. TSTRBUSY (bit 31) of EMACx\_TM\_NTGMT register indicates the status of synchronization. This enables user to program the start or stop time in advance, even before the earlier stop or start time has elapsed.

The start or stop time should be programmed with advanced system time to ensure proper PPS signal output. If the application programs a start or stop time that has already elapsed, then the EMAC sets TSTRGTERR (bit 3) in EMACx\_TM\_STMPSTAT register indicating the programming error. If enabled, the EMAC also sets the Target Time trigger (Alarm) interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

### PPS Width and Interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of sub-second increment value. For example, to have a PPS pulse width of 40ns and interval of 100ns, with the PTP reference clock of 50MHz, the width and interval should be programmed to values 2 and 5 respectively. Smaller granularity can be achieved by using a faster PTP reference clock. Before giving the command to trigger a pulse or pulse train on the PPS output, user should program or update the interval and width of the PPS signal output.

### PPS Command

When the PPS module is configured for Flexible PPS output, the PPSCTL (bits[3:0]) field in EMACx\_TM\_PPSCTL register can be used to command the PPS module for utilizing any of the Flexible PPS features.

Programming these bits with a non-zero value instructs the PPS module to initiate an event. Once the command is transferred or synchronized to the PTP clock domain, these bits gets cleared automatically. Software should ensure that these bits are programmed only when they are “all-zero”.

Table below explains the different commands available and their functionality.

| PPSCTL (bits [3:0]) | Command                      | Description                                                                                                                                                                                                                                                                                                                                                          |
|---------------------|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0000                | No Command                   |                                                                                                                                                                                                                                                                                                                                                                      |
| 0001                | Start Single Pulse           | Generates single pulse rising at start point defined in Target Time registers and of duration defined in EMACx_TM_PPSWIDTH register.                                                                                                                                                                                                                                 |
| 0010                | Start Pulse train            | Generates train of pulses rising at the Start Time configured in the Target Time registers, of duration configured in the EMACx_TM_PPSWIDTH register and repeated at interval configured in the EMACx_TM_PPSINTVL register. By default, the PPS pulse train is free-running unless stopped by 'Stop Pulse train at time' or 'Stop Pulse train immediately' commands. |
| 0011                | Cancel Start                 | Cancels the 'Start Single Pulse' and 'Start Pulse train' commands if the system time has not crossed the programmed start time.                                                                                                                                                                                                                                      |
| 0100                | Stop Pulse train at time     | Stops the train of pulses initiated by the 'Start Pulse train' command after the time programmed in the Target Time registers elapses.                                                                                                                                                                                                                               |
| 0101                | Stop Pulse train immediately | Immediately stops the train of pulses initiated by the 'Start Pulse train' command.                                                                                                                                                                                                                                                                                  |
| 0110                | Cancel Stop Pulse train      | Cancels the 'Stop Pulse train at time' command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.                                                                                                                                                                                    |
| 0111-1111           | Reserved                     |                                                                                                                                                                                                                                                                                                                                                                      |

### Interrupts related to PTP

Interrupts from PTP module can be enabled by setting TS (bit 9) in EMACx\_IMSK register. The status of the interrupt is indicated on TS (bit 9) of EMACx\_ISTAT register. PTP supports three types of interrupts.

### Auxiliary Snapshot Trigger

This interrupt is triggered when an external event occurs on PTP\_AUX\_IN pin and timestamp snapshot occurs. This is indicated on bit2 of EMACx\_TM\_STMPSTAT register.

### Target Time Reached

This interrupt is triggered when the System Time becomes equal to the value written in EMACx\_TM\_TGTM and EMACx\_TM\_NTGTM registers. It can be enabled or disabled by using TSTRIG (bit4) in EMACx\_TM\_CTL register and TRGTMODESEL (bit [6:5]) in EMACx\_TM\_PPSCTL register. This interrupt can be used as Alarm. This interrupt is indicated on TSTARGET (bit1) of EMACx\_TM\_STMPSTAT register.

## System Time Seconds Register Overflow

This interrupt is triggered when the EMACx\_TM\_SEC register overflows from 0xFFFF FFFF to 0x0000 0000. This interrupt is indicated on TSSOVF(bit 0) of EMACx\_TM\_STMPSTAT register. As soon as EMACx\_TM\_SEC register overflows, EMACx\_TM\_HISEC register increments by one.

## EMAC Event Control

The EMAC has a dedicated interrupt signal registered with the Processor System Event Controller. Various interrupt sources within EMAC peripheral are shared through this interrupt line. Please refer to the System Event Controller chapter for details on how interrupts work in BF60x and how to configure the same.

## EMAC Interrupt Signals

Interrupts from the EMAC can be triggered from the EMAC DMA layer or the EMAC CORE layer. Interrupts are triggered from EMAC DMA if a particular status bit is set in DMA status register. An interrupt line is asserted only when the corresponding bits are enabled in the DMA interrupt enable register. Similarly, interrupts are triggered from the EMAC CORE if a particular MMC status bit or PTP status bit is set in the interrupt status register. An interrupt line is asserted only when the corresponding bits are enabled in the MMC mask registers in case of MMC counters or the interrupt mask register in the case of PTP. Note that MMC interrupt status is also reflected in the DMA status register. The two groups of interrupts in the DMA status register are listed below.

NIS – Normal Interrupt source summary.

- Transmit Interrupt.
- Transmit Buffer Unavailable.
- Receive Interrupt.
- Early Receive Interrupt.

AIS – Abnormal Interrupt source summary.

- Transmit Process Stopped.
- Transmit Jabber Timeout.
- Receive FIFO Overflow.
- Transmit Underflow.
- Receive Buffer Unavailable.



- Receive Process Stopped.
- Receive Watchdog Timeout.
- Early Transmit Interrupt.
- Fatal Bus Error.

An interrupt is generated only once for simultaneous, multiple events. The driver must read the DMA Status Register for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in DMA status register. For example, the controller generates a Receive interrupt (bit 6 of DMA status register) and the driver begins reading DMA Status Register. Next, Receive Buffer Unavailable (bit 7 of DMA status register) occurs. The driver clears the Receive interrupt. Even then, the internal interrupt signal is not de-asserted, because of the active or pending receive buffer unavailable interrupt. Additionally, the driver must scan all descriptors, from the last recorded position to the first one owned by the DMA, in order to know which descriptor has asserted the interrupt.

Interrupts are cleared by writing a 1 to the corresponding bit position in the DMA status register. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared.

An interrupt delay timer is provided (receive interrupt watchdog timer register) is provided for flexible control of Receive Interrupt. When this Interrupt timer is programmed with a non-zero value, it gets activated as soon as the RxDMA completes a transfer of a received frame to system memory without asserting the Receive Interrupt because it is not enabled in the corresponding receive descriptor (RDES1[31] in Receive DMA descriptors). When this timer runs out (per the programmed value), the **RI** bit is set and the interrupt is asserted if the corresponding **RI** is enabled in the interrupt enable register. This timer is disabled before it runs out, when a frame is transferred to memory and **RI** is set because it is enabled for that descriptor.

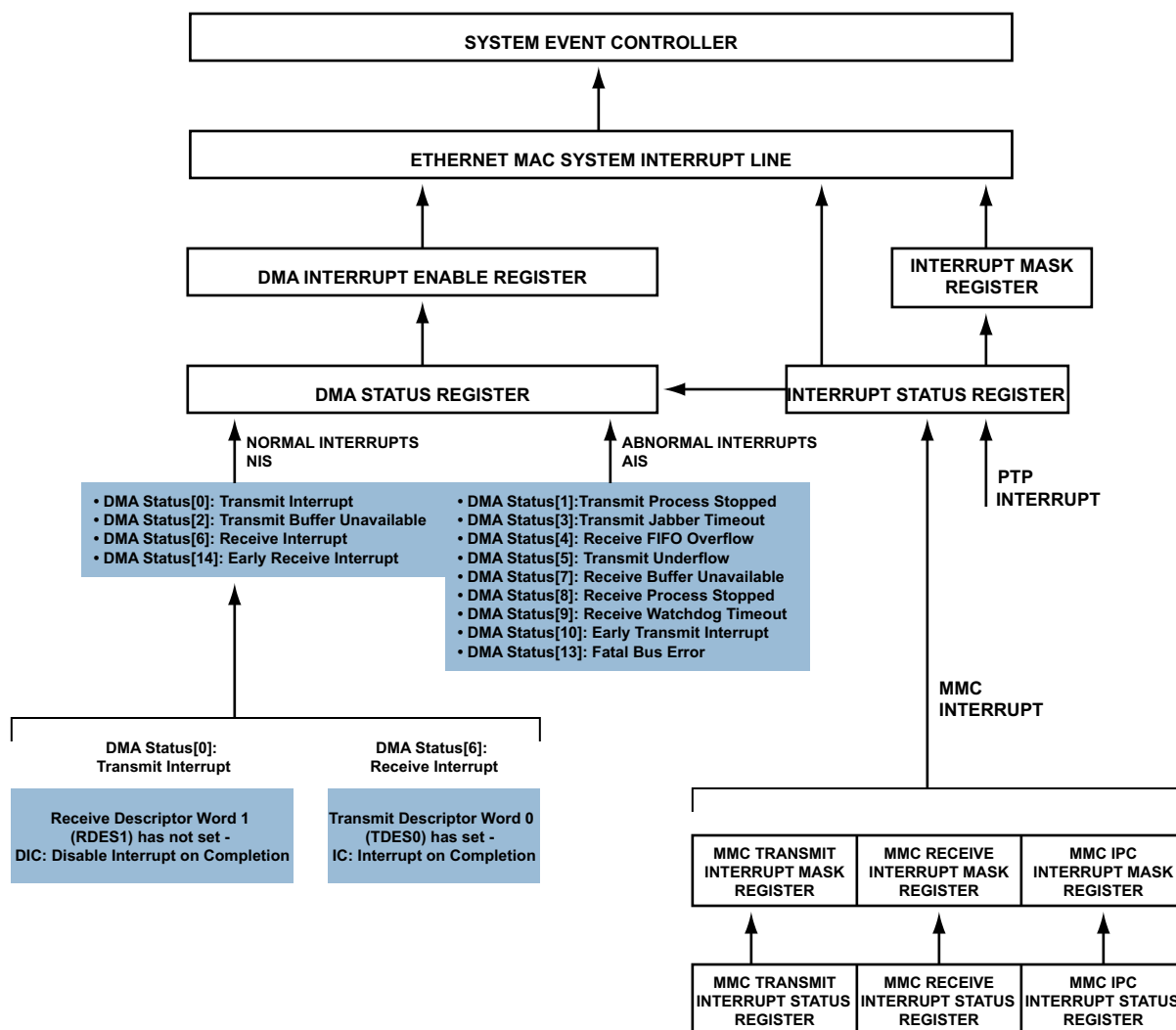


**Figure 23-15: EMAC Interrupt Flow Diagram**

## PHYINT Interrupt Signal

PHY device can notify the Blackfin EMAC when it detects changes to the link status, such as auto-negotiation or duplex-mode change. The external PHY chip typically includes an interrupt generation pin, in order to aid this status change notification to the MAC. This signal is typically termed as PHYINT and a falling/rising edge on this signal can be used to detect as a PHY interrupt at the EMAC.

In BF60x, any of the GPIO pin can be a PHYINT. In order to use the PHYINT, user may follow the below sequence:

1. Program the GPIO to detect a falling/rising edge sensitive interrupt.
2. Program the PHY to generate the Interrupt upon status change.
3. If PHYINT is asserted, read the PHY Status register via Station Management Interface.

**NOTE:** The PHYINT is not part of EMAC module, but rather any GPIO pin can be configured to interrupt the processor, upon detecting a rising edge generated by PHY.

Please refer to GPIO Chapter for more info on configuring GPIO pins for input.

## EMAC Programming Model

This section provides the programming model of Ethernet MAC peripheral for developers.

## EMAC Programming Steps

The following sections provide some general programming information

### DMA Initialization

1. Provide software reset by setting `SWR` bit in DMA Bus Mode Register (`DMA_BUSMODE`). This resets all of the EMAC internal registers and logic.
2. Wait for the completion of the reset process by polling `SWR` bit of the DMA Bus Mode Register (`DMA_BUSMODE`), which is only cleared (automatically) after the reset operation is completed.
3. Poll the `BUSRD` and `BUSWR` bits of the SCB Status Register (`DMA_BMSTAT`) to confirm that all previously initiated (before software-reset) or ongoing SCB transactions are complete.
4. Program the required fields in the DMA Bus Mode Register (`DMA_BUSMODE`):
  - a. Address Aligned Bursts.
  - b. Fixed burst or undefined burst.
  - c. Burst length values and burst mode values.
  - d. Descriptor Length (only valid if Ring Mode is used).
5. Program the SCB interface options in Bus Mode Register (`DMA_BMMODE`). If fixed burst-length is enabled, then select the maximum burst-length possible on the SCB bus (bits `BLN4`, `BLN8`, `BLN16`).
6. Create a proper descriptor chain for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (`OWN` bit of descriptor should be set). When `OSF` mode is used, at least two descriptors are required.
7. Make sure that your software creates three or more different transmit or receive descriptors in the chain before reusing any of the descriptors.

8. Initialize Receive Descriptor List Address Register (DMA\_RXDSC\_ADDR) and Transmit Descriptor List Address Register (DMA\_TXDSC\_ADDR) with the base address of receive and transmit descriptor respectively.
9. Program the required fields to initialize the mode of operation in Operation Mode Register (DMA\_OPMODE):
  - a. Receive and Transmit Store and Forward.
  - b. Receive and Transmit Threshold Control (RTC and TTC).
  - c. Error Frame and undersized good frame forwarding enable.
  - d. OSF Mode.
10. Clear the interrupt requests, by writing to those bits of the status register (interrupt bits only) that are set. For example, by writing 1 into bit 16, the normal interrupt summary clears this bit DMA Status Register (DMA\_STAT).
11. Enable the interrupts by programming the Interrupt Enable Register (DMA\_IEN).
12. Start the Receive and Transmit DMA by setting SR (bit 1) and ST (bit 13) of the DMA Operation Mode Register (DMA\_OPMODE).

## **EMAC CORE Initialization**

1. Program the EMAC Management Address Register (SMI\_ADDR) for controlling the management cycles for external PHY. For example, Physical Layer Address (bits 15-11). In addition, set bit 0 (SMI Busy) for writing into PHY and reading from PHY.
2. Read the 16-bit data of Management Data Register (SMI\_DATA) from the PHY for link up, speed of operation, and mode of operation, by specifying the appropriate address value in bits 15-11 of Management Address Register (SMI\_ADDR).
3. Program the MAC address in to EMAC Address0 High Register (ADDR0\_HI) and EMAC Address0 Low Register (ADDR0\_LO).
4. If Hash filtering is used, program Hash Table High Register (HASHTBL\_HI) and Hash Table Low Register (HASHTBL\_LO).

5. Program the required fields to set the appropriate filters for the incoming frames in MAC Frame Filter Register (MACFRMFILT):
  - a. Receive All.
  - b. Promiscuous mode.
  - c. Hash or Perfect Filter.
  - d. Unicast, multicast, broadcast, and control frames filter settings.
6. Program the required fields for proper flow control in Flow Control Register (FLOWCTL):
  - a. Pause time and other pause frame control bits.
  - b. Receive and Transmit Flow control bits.
  - c. Flow Control Busy/Backpressure Activate.
7. Program the EMAC Interrupt Mask Register bits (IMSK), as required.
8. Program the appropriate fields in MAC Configuration Register (MACCFG). For example, Inter-frame gap while transmission and jabber disable. Based on the Auto-negotiation you can set the Duplex mode (bit 11) or speed select (bit 14).
9. Set the bits Transmit enable (TE bit-3) and Receive Enable (RE bit-2) in MAC Configuration Register (MACCFG).

## Performing Normal Transmit and Receive Operations

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the application (either transmit or receive).
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and issuing a poll demand by writing 0 into the Tx/Rx poll demand register (DMA\_TXPOLL and DMA\_RXPOLL).
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process via Current Host Transmit Descriptor Register (DMA\_TXDSC\_CUR) and Current Host Receive Descriptor Register (DMA\_RXDSC\_CUR).
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process via Current Host Transmit Buffer Address Register (DMA\_TXBUF\_CUR) and Current Host Receive Buffer Address Register (DMA\_RXBUF\_CUR).

## Stopping and Starting Transfers

1. Disable the Transmit DMA (if applicable), by clearing bit 13 (ST: Start/Stop Transmission Command) of DMA Operation Mode Register (DMA\_OPMODE).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of EMAC Debug Register (DBG).
3. Disable the MAC transmitter and MAC receiver by clearing the bit 3 (TE: Transmitter Enable) and bit 2 (RE: Receiver Enable) in MAC Configuration Register (MACCFG).
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory by reading EMAC Debug Register (DBG).
5. Make sure that both Tx FIFO and Rx FIFO are empty.
6. To re-start the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

## EMAC Interrupts and ISRs

Specific steps for enabling interrupts and using their ISRs are outlined here.

*PREREQUISITE:* This may typically be assumed on top of steps for EMAC and DMA initialization and operations.

1. Receive interrupts are enabled for descriptors by default. Transmit interrupts must be enabled for individual descriptors by setting bit IC (bit 30) in TDES0 word of transmit descriptor.
2. Enable required bits in DMA Interrupt Enable register DMA\_IEN. Setting AIE or NIE can turn on the occurrence of all normal / abnormal interrupt conditions. Different conditions may also be turned on directly by setting individual bits in DMA\_IEN register.
3. MMC overflow interrupts are enabled by setting appropriate bits in MMC\_TXIMSK and MMC\_RXIMSK registers.
4. PTP interrupts are enabled by setting TS bit in EMACx\_IMSK register.

5. Once EMAC interrupt is asserted and SEC branches execution to EMAC ISR, below sequence may be followed in software:
  - a. Read DMA status from `DMA_STAT` register.
  - b. Clear interrupt source by writing 1 in to the bits that are set in `DMA_STAT` (W1C).
  - c. Check for normal / abnormal / mmc / ptp interrupt by parsing status bits in DMA status value read earlier, and call appropriate service function.
  - d. Typical normal interrupt assertions include Transmit and Receive Interrupt.
  - e. Typical abnormal interrupt assertion include Receive Underflow.
6. MMC handler functions may follow the below sequence:
  - a. First read whether it is Transmit Counter Interrupt or Receive Counter Interrupt, by reading `ISTAT` register and parsing for `MMCTX` and `MMCRX` bits respectively.
  - b. Read `MMC_RXINT` or `MMC_TXINT` registers to check for which of the counters have triggered interrupt.
  - c. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.
7. PTP handler functions may follow the below sequence:
  - a. First read whether it is PTP Interrupt, by reading `ISTAT` register and parsing for `TS` bit.
  - b. Read `TM_STMPSTAT` register and determine interrupt source by parsing `ATSTS`, `TSTARGT` and `TSSOVF` bits.
  - c. Interrupt source is cleared by reading `TM_STMPSTAT` register.

## Enabling Checksum for Transmit and Receive

Specific steps for enabling transmit and receive checksum are outlined here.

### *PREREQUISITE:*

This may typically be assumed on top of steps for EMAC and DMA initialization and operations. Note that transmit and receive checksum features are independent of each other.

1. To enable Transmit Checksum Insertion:
  - a. Enable Store Forward mode in FIFO (set `TSF` bit in `OPMODE` register).
  - b. Make sure that the transmit frame can be contained within the TX FIFO of 256 bytes confirming to the size rule:  $\text{FIFO Depth} - \text{PBL} - 3 \text{ FIFO Locations}$ , where PBL is Burst Length.
  - c. Program required option for transmit checksum, by programming (CIC) Checksum Insertion Control in Transmit Descriptor Word 0 (TDES0): IP header checksum / IP header checksum and payload checksum / IP Header checksum, payload checksum and pseudo header checksum.
  - d. Higher layer such as IP Stack sends out packet to EMAC which inserts checksum as configured above.
2. To enable Receive Checksum Verification:
  - a. Enable Receive Checksum off-load engine (set `IPC` bit in `MACCFG` register).
  - b. Enable 8 word descriptor (32 bytes), by setting `ATDS` bit in DMA Bus Mode register.
  - c. Provide a total of 8 x 32-bit word space for receive descriptor.
  - d. Wait for Receive Interrupt and check for extended status availability by parsing bit 0 in RDES0 word.
  - e. If extended status available, read RDES4 and pass to higher layer such as IP-stack.
  - f. Higher software layer may check for IPv4/IPv6/Payload Type and Checksum Payload/Header errors.

## Programming System Time Module

*PREREQUISITE:* PTP configurations and programming may typically be assumed on top of steps for EMAC and DMA initialization and operations., since PTP is a sub-block in EMAC.

1. Enable PTP module by setting `TSENA` (bit 0) in `EMACx_TM_CTL` register.



## 2. System Time Initialization

- a. The time (seconds and nanoseconds) at which System Time should be initialized should be written into EMACx\_TM\_SECUPDT and EMACx\_TM\_NSECUPDT registers.
- b. Set TSINIT (bit2) in EMACx\_TM\_CTL register. System Time is initialized and TSINIT bit clears automatically.
- c. Binary or Digital rollover of EMACx\_TM\_NSEC register can be chosen by configuring TSCTRLSSR (bit9) in EMACx\_TM\_CTL register.

## 3. System Time Coarse Correction

- a. The Offset time (seconds and nanoseconds) which has to be added or subtracted to System Time, should be written into EMACx\_TM\_SECUPDT and EMACx\_TM\_NSECUPDT registers.
- b. Configure ADDSUB (bit31) of EMACx\_TM\_NSECUPDT register to choose between add or subtract offset time.
- c. Set TSUPDT (bit3) in EMACx\_TM\_CTL register. System Time is corrected with offset time and TSUPDT bit clears automatically.

## 4. System Time Fine Correction

- a. Calculate required Addend value based on input PTP clock frequency and required frequency. Refer to Fine Correction Method section.
- b. Write the calculated Addend value in EMACx\_TM\_ADDEND register and set TSADDREG (bit5) in EMACx\_TM\_CTL register. Addend value is updated and TSADDREG bit is cleared automatically.
- c. Configure EMACx\_TM\_SUBSEC register based on new PTP frequency.

## 5. Target Time trigger (Alarm)

- a. Set TS (bit 9) in EMACx\_IMSK register to enable PTP interrupts.
- b. Program TRGTMODESEL (bits[6:5]) in EMACx\_TM\_PPSCTL register with 00 or 10 (for PPS start/stop time programming).
- c. Program the time when interrupt should be triggered, in the EMACx\_TM\_TGTM and EMACx\_TM\_NTGTM registers. The programmed time should be greater than the current System Time. Else Target time programming error occurs indicated by TSTRGTERR (bit 3) in EMACx\_TM\_STMPSTAT register.
- d. Set TSTRIG (bit 4) in EMACx\_TM\_CTL register to enable Target Time trigger interrupt. After System Time reaches programmed target time (in step 2), Target time trigger interrupt occurs indicated by TSTARGET (bit 1) in EMACx\_TM\_STMPSTAT register and TS (bit 9) in EMACx\_ISTAT register. TSTRIG is cleared automatically.

## Programming PTP module for Frame detection and Timestamping

*PREREQUISITE:* PTP configurations and programming may typically be assumed on top of steps for EMAC and DMA initialization and operations., since PTP is a sub-block in EMAC.

1. For timestamping a transmitting frame, TTSE bit in TDES0 of corresponding frame should be set.
2. Extend the descriptor word length from 4 WORDS to 8 WORDS by setting ATDS (bit 7) in EMACx\_DMA\_BUSMODE register.
3. Configure the fields (bit [18:10]) in EMACx\_TM\_CTL register so that PTP module will detect and/or time-stamp only specific types of received frames. Refer to EMACx\_TM\_CTL register description for more information.
4. Select PTP clock source by programming PADS0\_EMAC\_PTP\_CLKSEL register.
5. PTP module should be enabled by setting TSENA (bit 0) in EMACx\_TM\_CTL register.
6. System Time should be initialized.
7. Verify the RDES4 for status of the received frame and RDES6 and RDES7 for Timestamp nanoseconds and seconds value.

## Programming for Auxiliary Timestamps

*PREREQUISITE:* PTP configurations and programming may typically be assumed on top of steps for EMAC and DMA initialization and operations., since PTP is a sub-block in EMAC.

1. Set TS (bit 9) in EMACx\_IMSK register to enable PTP interrupts.
2. PTP module should be enabled by setting TSENA (bit 0) in EMACx\_TM\_CTL register.
3. System Time should be initialized.
4. Whenever a rising edge on Auxiliary Timestamp Trigger pin is detected, System Time seconds and nanoseconds are captured and stored into 4-deep Auxiliary Timestamp FIFO. Auxiliary timestamp trigger interrupt occurs indicated by ATSTS (bit 2) in EMACx\_TM\_STMPSTAT register and TS (bit 9) in EMACx\_ISTAT register.
5. The contents of FIFO can be read one by one through EMACx\_TM\_AUXSTMP\_SEC and EMACx\_TM\_AUXSTMP\_NSEC registers. One level of FIFO is cleared when EMACx\_TM\_AUXSTMP\_SEC register is read. Hence read EMACx\_TM\_AUXSTMP\_NSEC register before EMACx\_TM\_AUXSTMP\_SEC register.
6. FIFO can be cleared by setting ATSFC (bit24) in EMACx\_TM\_CTL register.

## Programming Pulse-Per-Second Output

*PREREQUISITE:* PTP configurations and programming may typically be assumed on top of steps for EMAC and DMA initialization and operations., since PTP is a sub-block in EMAC.

### 1. Fixed Pulse-Per-Second Output

- a. PTP module should be enabled by setting TSENA (bit 0) in EMACx\_TM\_CTL register.
- b. Configure PPSCTL (bits [3:0]) in EMACx\_TM\_PPSCTL register and Binary or Digital rollover by configuring TSCTRLSSR (bit 9) in EMACx\_TM\_CTL register, so as to output required PPS waveform. Refer to Fixed PPS section.

### 2. Flexible Pulse-Per-Second Output

- a. PTP module should be enabled by setting TSENA (bit 0) in EMACx\_TM\_CTL register.
- b. Set PPSEN (bit 4) of EMACx\_TM\_PPSCTL register to enable Flexible PPS output.
- c. Program TRGTMODESEL (bits[6:5]) in EMACx\_TM\_PPSCTL register with 11 or 10 (for Target time trigger interrupt).
- d. Program the start time value when PPS output should start, in the EMACx\_TM\_TGTM and EMACx\_TM\_NTGTM registers. Ensure that TSTRBUSY (bit 31) of EMACx\_TM\_NTGTM register is reset before programming the Target Time registers again.
- e. Program the period of PPS signal output in EMACx\_TM\_PPSINTVL register for Pulse Train output, and width of the PPS signal output in EMACx\_TM\_PPSWIDTH register for Single Pulse or Pulse Train output.
- f. Ensure that PPSCTL (bits [3:0]) of EMACx\_TM\_PPSCTL register is cleared and program PPSCTL to 0001 to start Single Pulse, or program PPSCTL to 0010 to start Pulse Train at programmed start time (step 4). The PPS Pulse Train is free-running unless stopped by 'STOP Pulse train at time Command' (PPSCTL=0100) or 'STOP Pulse Train immediately Command' (PPSCTL=0101).
- g. Start of Pulse generation can be cancelled by giving the 'Cancel Start Command' (PPSCTL=0011) before the programmed start time (step 4) elapses.
- h. Program the stop time value when PPS output should stop, in the EMACx\_TM\_TGTM and EMACx\_TM\_NTGTM registers. Ensure that TSTRBUSY (bit 31) of EMACx\_TM\_NTGTM register is reset before programming the Target Time registers again.
- i. Ensure that PPSCTL (bits [3:0]) of EMACx\_TM\_PPSCTL register is cleared and program PPSCTL to 0100. This stops the train of pulses on PPS signal output after the programmed stop time (step 8) elapses.
- j. Pulse train can be stopped immediately by giving 'STOP Pulse Train immediately Command' (PPSCTL=0101). Similarly, 'Stop Pulse train Command' (given in step 9) can be cancelled by programming 0110 in the PPSCTL field before the programmed stop time (step 8) elapses.

## EMAC Programming Concepts

### IEEE 802.3 Ethernet Packet Structure

Typical frame format of an Ethernet packet is given below. Please refer to IEEE standards for detailed information on Ethernet packets and their format.

**Table 23-36:** IEEE 802.3 frame structure

| Parameter<br>Description (original source: <a href="http://www.wikipedia.com">www.wikipedia.com</a> ).                                                                                                                                                                                                                        | Position in Ethernet<br>packet | Total Bytes |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|-------------|
| <b>PREAMBLE</b><br>The preamble of an Ethernet frame consists of a 56-bit (7-byte) pattern of alternating 1 and 0 bits (#10101010), which allows devices on the network to easily detect a new incoming frame (synchronization purpose).                                                                                      | 1                              | 7           |
| <b>SFD</b><br>The SFD (#10101011) is a 1-byte pattern designed to break the preamble pattern, and signal the start of the actual frame.                                                                                                                                                                                       | 2                              | 1           |
| <b>DA</b><br>48 bits long Destination address. This can be a Unicast, Multicast or Broadcast address.                                                                                                                                                                                                                         | 3                              | 6           |
| <b>SA</b><br>48 bits long Source address. This is typically Unicast, Multicast or Broadcast address.                                                                                                                                                                                                                          | 4                              | 6           |
| <b>LT</b><br>Typically this field is the length in terms of the number of bytes and can be anywhere between 0 – 1500. When the value is greater than or equal to 0x0600, this field is also used to indicate the type of special payload carried by the frame. Examples include: 0x8808 for flow control and 0x0800 for IPv4. | 5                              | 2           |
| <b>DATA</b><br>Actual application data payload; usually between 0 – 1500.                                                                                                                                                                                                                                                     | 6                              | 0 - 1500    |
| <b>PAD</b><br>Frames must be at least 64 bytes long, not including the preamble, so, if the Data field is shorter than 46 bytes, it must be compensated by the Pad field.                                                                                                                                                     | 7                              | 0 - 46      |

**Table 23-36:** IEEE 802.3 frame structure (Continued)

| Parameter<br>Description (original source: www.wikipedia.com).                                                                                                                                                                                                                                                               | Position in Ethernet<br>packet | Total Bytes |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|-------------|
| FCS<br>The frame check sequence is a 32-bit cyclic redundancy check which enables detection of corrupted data within the entire frame. This is generated from a CRC-32 polynomial code (CRC-32-IEEE): $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ . | 8                              | 4           |

## Frame Size Statistics for Application Software

**Table 23-37:** Ethernet Frame Size statistics

| Frame size statistics        | VLAN specific change<br>Comments                                                                    |                                                                                   |
|------------------------------|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|
| Information bytes / Header   | 4 byte 802.1Q header inserted after Source Address and before Type/LAN in 802.3 packets = 22 bytes. | $6 \times 2 + 2 + 4 = 18$ bytes (DA+SA+LT+FCS)                                    |
| Minimum Frame Size (typical) | If DATA is NULL, 42 byte padding is done to make 64 bytes (42 + 22)                                 | 64 bytes. If DATA is NULL, 46 byte padding is done to make 64 bytes (46 + 18)     |
| Maximum Frame Size (typical) | 1522 bytes                                                                                          | 1518 bytes (1500 bytes DATA and 18 bytes header)                                  |
| Jumbo Frame Size             | 9022 bytes                                                                                          | Typical industry standard Ethernet jumbo frame size may be treated as 9018 bytes. |

## Software Visualization of Programmable Packet Size

**Table 23-38:** Visualization of Programmable Packet Size

| Size in bytes | Comments                                                            |
|---------------|---------------------------------------------------------------------|
| 16384         | Receive Watchdog & Transmit Jabber disabled, Jumbo frames enabled.  |
| 10240         | Receive Watchdog & Transmit Jabber disabled, Jumbo frames disabled. |
| 2048          | Receive Watchdog & Transmit Jabber enabled.                         |

**Table 23-38:** Visualization of Programmable Packet Size (Continued)

| Size in bytes | Comments                                                                          |
|---------------|-----------------------------------------------------------------------------------|
| 1518          | Typical max size of Ethernet frame. Receive Watchdog and Transmit Jabber enabled. |
| 64            | Typical min size of Ethernet frame.                                               |
| < 64          | Runt frames requiring Zero-PAD.                                                   |

## Ethernet Packet Structure in C

```
/*Example for Ethernet packet structure in C language*/
typedef struct ETHER_PACKET
{
    char  dst_addr[6];           //destination address
    char  src_addr[6];           //source address
    char  length[2];             //length of actual data
    char  data[DATA_SIZE];       //application data
    char  fdlimit[DELIMIT_SIZE]; //32-bit delimit (if manual appending)
    char  fcs[4];                //crc frame checksum, used by RX buffer.
} ETHER_PACKET;
```

## DMA Descriptor Implementation in C

```
/*
 * Following is a simple implementation of Descriptors in Ring and Chain model in C
 language.
 * Typically 4 WORDs (32-bit) are used for descriptors. Using Checksum offload or PTP
 Engine requires 8 WORDs.
 * Only high-level common functionalities across transmit and receive descriptors are
 considered here.
 */
/* DMA Ring Descriptor */
typedef struct EMAC_DMADESC_RING
{
    unsigned int      Status; //TDES0 OR RDES0
    unsigned int      ControlDesc; //TDES1 OR RDES1
    unsigned int      StartAddr1; //TDES2 OR RDES2
    unsigned int      StartAddr2; //TDES3 OR RDES3
#ifdef CHECKSUM_OFFLOAD
    struct EMAC_EXT_STAT ExtendedStat;
#endif
} EMAC_DMADESC_RING;
/* DMA Chain Descriptor */
typedef struct EMAC_DMADESC_CHAIN
{
    unsigned int      Status; //TDES0 OR RDES0
    unsigned int      ControlDesc; //TDES1 OR RDES1
```

```

        unsigned int                StartAddr; //TDES2 OR RDES2
        struct EMAC_DMADESC_CHAIN *pNextDesc; //TDES3 OR RDES3
#ifdef CHECKSUM_OFFLOAD
        struct EMAC_EXT_STAT ExtendedStat;
#endif
    } EMAC_DMADESC_CHAIN;
/* Extended Status Descriptor with PTP not enabled*/
typedef struct EMAC_EXT_STAT
{
    #ifdef RX_DESC
        unsigned int                CheckSumStat; //RDES4
    #ifdef TX_DESC
        unsigned int                Reserved; //TDES4
    #endif
        unsigned int                Reserved; //RDES5 OR TDES5
        unsigned int                Reserved; //RDES6 OR TDES6
        unsigned int                Reserved; //RDES7 OR TDES7
    } EMAC_EXT_STAT;

```

## PTP Header Structure in C

```

/* PTP Message Format (Refer to Table23-33)*/
typedef struct EMAC_PTP_HEADER
{
    unsigned char messageType:4; //PTP Version 2 message type
    transportSpecific:4;
    unsigned char versionPTP; //PTP Version (1 or 2)
    unsigned short messageLength;
    unsigned char domainNumber;
    unsigned char RESERVED1;
    unsigned short flagField;
    unsigned char correctionField[8];
    unsigned char RESERVED2[4];
    unsigned char sourcePortIdentity[10];
    unsigned short sequenceId;
    unsigned char controlField; //PTP Version 1 message type
    unsigned char logMessageInterval;
} EMAC_PTP_HEADER;

```

## EMAC Register Descriptions

Ethernet MAC (EMAC) contains the following registers.

**Table 23-39:** EMAC Module Registers

| Name        | Description                |
|-------------|----------------------------|
| EMAC_MACCFG | MAC Configuration Register |

**Table 23-39:** EMAC Module Registers (Continued)

| Name              | Description                           |
|-------------------|---------------------------------------|
| EMAC_MACFRMFILT   | MAC Rx Frame Filter Register          |
| EMAC_HASHTBL_HI   | Hash Table High Register              |
| EMAC_HASHTBL_LO   | Hash Table Low Register               |
| EMAC_SMI_ADDR     | SMI Address Register                  |
| EMAC_SMI_DATA     | SMI Data Register                     |
| EMAC_FLOWCTL      | FLow Control Register                 |
| EMAC_VLANTAG      | VLAN Tag Register                     |
| EMAC_DBG          | Debug Register                        |
| EMAC_ISTAT        | Interrupt Status Register             |
| EMAC_IMSK         | Interrupt Mask Register               |
| EMAC_ADDR0_HI     | MAC Address 0 High Register           |
| EMAC_ADDR0_LO     | MAC Address 0 Low Register            |
| EMAC_MMC_CTL      | MMC Control Register                  |
| EMAC_MMC_RXINT    | MMC Rx Interrupt Register             |
| EMAC_MMC_TXINT    | MMC Tx Interrupt Register             |
| EMAC_MMC_RXIMSK   | MMC Rx Interrupt Mask Register        |
| EMAC_MMC_TXIMSK   | MMC TX Interrupt Mask Register        |
| EMAC_TXOCTCNT_GB  | Tx OCT Count (Good/Bad) Register      |
| EMAC_TXFRMCNT_GB  | Tx Frame Count (Good/Bad) Register    |
| EMAC_TXBCASTFRM_G | Tx Broadcast Frames (Good) Register   |
| EMAC_TXMCASTFRM_G | Tx Multicast Frames (Good) Register   |
| EMAC_TX64_GB      | Tx 64-Byte Frames (Good/Bad) Register |



**Table 23-39:** EMAC Module Registers (Continued)

| Name                | Description                                     |
|---------------------|-------------------------------------------------|
| EMAC_TX65T0127_GB   | Tx 65- to 127-Byte Frames (Good/Bad) Register   |
| EMAC_TX128T0255_GB  | Tx 128- to 255-Byte Frames (Good/Bad) Register  |
| EMAC_TX256T0511_GB  | Tx 256- to 511-Byte Frames (Good/Bad) Register  |
| EMAC_TX512T01023_GB | Tx 512- to 1023-Byte Frames (Good/Bad) Register |
| EMAC_TX1024TOMAX_GB | Tx 1024- to Max-Byte Frames (Good/Bad) Register |
| EMAC_TXUCASTFRM_GB  | Tx Unicast Frames (Good/Bad) Register           |
| EMAC_TXMCASTFRM_GB  | Tx Multicast Frames (Good/Bad) Register         |
| EMAC_TXBCASTFRM_GB  | Tx Broadcast Frames (Good/Bad) Register         |
| EMAC_TXUNDR_ERR     | Tx Underflow Error Register                     |
| EMAC_TXSNGCOL_G     | Tx Single Collision (Good) Register             |
| EMAC_TXMULTCOL_G    | Tx Multiple Collision (Good) Register           |
| EMAC_TXDEFERRED     | Tx Deferred Register                            |
| EMAC_TXLATECOL      | Tx Late Collision Register                      |
| EMAC_TXEXCESSCOL    | Tx Excess Collision Register                    |
| EMAC_TXCARR_ERR     | Tx Carrier Error Register                       |
| EMAC_TXOCTCNT_G     | Tx Octet Count (Good) Register                  |
| EMAC_TXFRMCNT_G     | Tx Frame Count (Good) Register                  |
| EMAC_TXEXCESSDEF    | Tx Excess Deferral Register                     |
| EMAC_TXPAUSEFRM     | Tx Pause Frame Register                         |
| EMAC_TXVLANFRM_G    | Tx VLAN Frames (Good) Register                  |
| EMAC_RXFRMCNT_GB    | Rx Frame Count (Good/Bad) Register              |
| EMAC_RXOCTCNT_GB    | Rx Octet Count (Good/Bad) Register              |

**Table 23-39:** EMAC Module Registers (Continued)

| Name                | Description                                     |
|---------------------|-------------------------------------------------|
| EMAC_RXOCTCNT_G     | Rx Octet Count (Good) Register                  |
| EMAC_RXBCASTFRM_G   | Rx Broadcast Frames (Good) Register             |
| EMAC_RXMCASTFRM_G   | Rx Multicast Frames (Good) Register             |
| EMAC_RXCRC_ERR      | Rx CRC Error Register                           |
| EMAC_RXALIGN_ERR    | Rx alignment Error Register                     |
| EMAC_RXRUNT_ERR     | Rx Runt Error Register                          |
| EMAC_RXJAB_ERR      | Rx Jab Error Register                           |
| EMAC_RXUSIZE_G      | Rx Undersize (Good) Register                    |
| EMAC_RXOSIZE_G      | Rx Oversize (Good) Register                     |
| EMAC_RX64_GB        | Rx 64-Byte Frames (Good/Bad) Register           |
| EMAC_RX65T0127_GB   | Rx 65- to 127-Byte Frames (Good/Bad) Register   |
| EMAC_RX128T0255_GB  | Rx 128- to 255-Byte Frames (Good/Bad) Register  |
| EMAC_RX256T0511_GB  | Rx 256- to 511-Byte Frames (Good/Bad) Register  |
| EMAC_RX512T01023_GB | Rx 512- to 1023-Byte Frames (Good/Bad) Register |
| EMAC_RX1024TOMAX_GB | Rx 1024- to Max-Byte Frames (Good/Bad) Register |
| EMAC_RXUCASTFRM_G   | Rx Unicast Frames (Good) Register               |
| EMAC_RXLEN_ERR      | Rx Length Error Register                        |
| EMAC_RXOORTYPE      | Rx Out Of Range Type Register                   |
| EMAC_RXPAUSEFRM     | Rx Pause Frames Register                        |
| EMAC_RXFIFO_OVF     | Rx FIFO Overflow Register                       |
| EMAC_RXVLANFRM_GB   | Rx VLAN Frames (Good/Bad) Register              |
| EMAC_RXWDG_ERR      | Rx Watch Dog Error Register                     |

**Table 23-39:** EMAC Module Registers (Continued)

| Name                    | Description                                    |
|-------------------------|------------------------------------------------|
| EMAC_IPC_RXIMSK         | MMC IPC Rx Interrupt Mask Register             |
| EMAC_IPC_RXINT          | MMC IPC Rx Interrupt Register                  |
| EMAC_RXIPV4_GD_FRM      | Rx IPv4 Datagrams (Good) Register              |
| EMAC_RXIPV4_HDR_ERR_FRM | Rx IPv4 Datagrams Header Errors Register       |
| EMAC_RXIPV4_NOPAY_FRM   | Rx IPv4 Datagrams No Payload Frame Register    |
| EMAC_RXIPV4_FRAG_FRM    | Rx IPv4 Datagrams Fragmented Frames Register   |
| EMAC_RXIPV4_UDSBL_FRM   | Rx IPv4 UDP Disabled Frames Register           |
| EMAC_RXIPV6_GD_FRM      | Rx IPv6 Datagrams Good Frames Register         |
| EMAC_RXIPV6_HDR_ERR_FRM | Rx IPv6 Datagrams Header Error Frames Register |
| EMAC_RXIPV6_NOPAY_FRM   | Rx IPv6 Datagrams No Payload Frames Register   |
| EMAC_RXUDP_GD_FRM       | Rx UDP Good Frames Register                    |
| EMAC_RXUDP_ERR_FRM      | Rx UDP Error Frames Register                   |
| EMAC_RXTCP_GD_FRM       | Rx TCP Good Frames Register                    |
| EMAC_RXTCP_ERR_FRM      | Rx TCP Error Frames Register                   |
| EMAC_RXICMP_GD_FRM      | Rx ICMP Good Frames Register                   |
| EMAC_RXICMP_ERR_FRM     | Rx ICMP Error Frames Register                  |
| EMAC_RXIPV4_GD_OCT      | Rx IPv4 Datagrams Good Octets Register         |
| EMAC_RXIPV4_HDR_ERR_OCT | Rx IPv4 Datagrams Header Errors Register       |
| EMAC_RXIPV4_NOPAY_OCT   | Rx IPv4 Datagrams No Payload Octets Register   |
| EMAC_RXIPV4_FRAG_OCT    | Rx IPv4 Datagrams Fragmented Octets Register   |
| EMAC_RXIPV4_UDSBL_OCT   | Rx IPv4 UDP Disabled Octets Register           |
| EMAC_RXIPV6_GD_OCT      | Rx IPv6 Good Octets Register                   |

**Table 23-39:** EMAC Module Registers (Continued)

| Name                    | Description                                   |
|-------------------------|-----------------------------------------------|
| EMAC_RXIPV6_HDR_ERR_OCT | Rx IPv6 Header Errors Register                |
| EMAC_RXIPV6_NOPAY_OCT   | Rx IPv6 No Payload Octets Register            |
| EMAC_RXUDP_GD_OCT       | Rx UDP Good Octets Register                   |
| EMAC_RXUDP_ERR_OCT      | Rx UDP Error Octets Register                  |
| EMAC_RXTCP_GD_OCT       | Rx TCP Good Octets Register                   |
| EMAC_RXTCP_ERR_OCT      | Rx TCP Error Octets Register                  |
| EMAC_RXICMP_GD_OCT      | Rx ICMP Good Octets Register                  |
| EMAC_RXICMP_ERR_OCT     | Rx ICMP Error Octets Register                 |
| EMAC_TM_CTL             | Time Stamp Control Register                   |
| EMAC_TM_SUBSEC          | Time Stamp Sub Second Increment Register      |
| EMAC_TM_SEC             | Time Stamp Low Seconds Register               |
| EMAC_TM_NSEC            | Time Stamp Nano Seconds Register              |
| EMAC_TM_SECUPDT         | Time Stamp Seconds Update Register            |
| EMAC_TM_NSECUPDT        | Time Stamp Nano Seconds Update Register       |
| EMAC_TM_ADDEND          | Time Stamp Addend Register                    |
| EMAC_TM_TGTM            | Time Stamp Target Time Seconds Register       |
| EMAC_TM_NTGTM           | Time Stamp Target Time Nano Seconds Register  |
| EMAC_TM_HISEC           | Time Stamp High Second Register               |
| EMAC_TM_STMPSTAT        | Time Stamp Status Register                    |
| EMAC_TM_PPSCTL          | PPS Control Register                          |
| EMAC_TM_AUXSTMP_NSEC    | Time Stamp Auxiliary TS Nano Seconds Register |
| EMAC_TM_AUXSTMP_SEC     | Time Stamp Auxiliary TM Seconds Register      |

**Table 23-39:** EMAC Module Registers (Continued)

| Name                | Description                             |
|---------------------|-----------------------------------------|
| EMAC_TM_PPSINTVL    | Time Stamp PPS Interval Register        |
| EMAC_TM_PPSWIDTH    | PPS Width Register                      |
| EMAC_DMA_BUSMODE    | DMA Bus Mode Register                   |
| EMAC_DMA_TXPOLL     | DMA Tx Poll Demand Register             |
| EMAC_DMA_RXPOLL     | DMA Rx Poll Demand register             |
| EMAC_DMA_RXDSC_ADDR | DMA Rx Descriptor List Address Register |
| EMAC_DMA_TXDSC_ADDR | DMA Tx Descriptor List Address Register |
| EMAC_DMA_STAT       | DMA Status Register                     |
| EMAC_DMA_OPMODE     | DMA Operation Mode Register             |
| EMAC_DMA_IEN        | DMA Interrupt Enable Register           |
| EMAC_DMA_MISS_FRM   | DMA Missed Frame Register               |
| EMAC_DMA_RXIWDG     | DMA Rx Interrupt Watch Dog Register     |
| EMAC_DMA_BMODE      | DMA SCB Bus Mode Register               |
| EMAC_DMA_BMSTAT     | DMA SCB Status Register                 |
| EMAC_DMA_TXDSC_CUR  | DMA Tx Descriptor Current Register      |
| EMAC_DMA_RXDSC_CUR  | DMA Rx Descriptor Current Register      |
| EMAC_DMA_TXBUF_CUR  | DMA Tx Buffer Current Register          |
| EMAC_DMA_RXBUF_CUR  | DMA Rx Buffer Current Register          |

## MAC Configuration Register

The EMAC\_MACCFG register configures MAC features.

EMAC\_MACCFG: MAC Configuration Register - R/W

Reset = 0x0000 8000

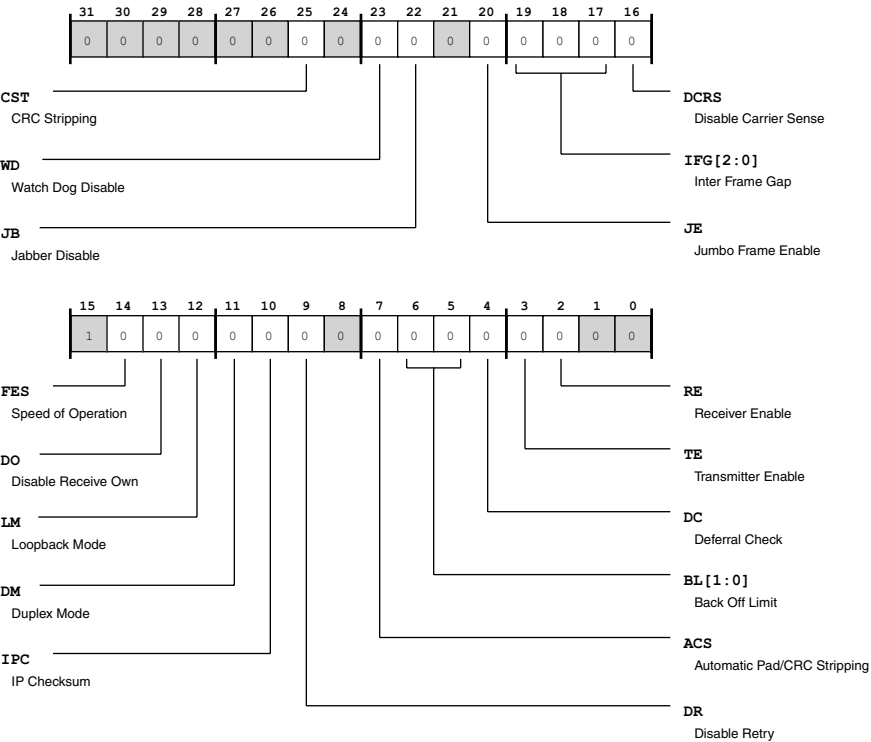


Figure 23-16: EMAC\_MACCFG Register Diagram

Table 23-40: EMAC\_MACCFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | CST      | CRC Stripping.<br>The EMAC_MACCFG.CST bit, when set, directs the MAC to strip the last 4 bytes (FCS) of all frames of Ether type (type field greater than 0x0600) and drop these bytes before forwarding the frame to the application.                                                                               |
| 23      | WD       | Watch Dog Disable.<br>The EMAC_MACCFG.WD bit when set, disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is reset, the MAC allows no more than 2,048 bytes (10,240 if JE is set high) of the frame being received and cuts off any bytes received after that. |

**Table 23-40:** EMAC\_MACCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | JB       | Jabber Disable.<br>The EMAC_MACCFG.JB bit when set, disables the jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.                                                                                |
| 20      | JE       | Jumbo Frame Enable.<br>The EMAC_MACCFG.JE bit when set, direct the MAC to allow Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.                                                                                                                                                                                 |
| 19:17   | IFG      | Inter Frame Gap.<br>The EMAC_MACCFG.IFG bits control the minimum IFG between frames during transmission. Note that in Half-Duplex mode, the minimum IFG can be configured for 64 bit times (IFG = 100) only. Lower values are not considered.                                                                                                                                                       |
|         |          | 0                      96 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1                      88 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 2                      80 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 3                      72 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 4                      64 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 5                      56 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 6                      48 bit times                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7                      40 bit times                                                                                                                                                                                                                                                                                                                                                                 |
| 16      | DCRS     | Disable Carrier Sense.<br>The EMAC_MACCFG.DCRS bit, when set, makes the MAC transmitter ignore the CRS signal during frame transmission in Half-Duplex mode. This request results in no errors generated due to Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors due to Carrier Sense and will even abort the transmissions. |

**Table 23-40:** EMAC\_MACCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | FES      | Speed of Operation.<br>The EMAC_MACCFG.FES bit indicates the Ethernet speed as: 0= 10 Mbps and 1= 100 Mbps.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 13      | DO       | Disable Receive Own.<br>The EMAC_MACCFG.DO bit, when set, disables MAC reception of frames when it is transmitting in Half-Duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in Full-Duplex mode.                                                                                                                                                                                                                                                                                                                                                        |
| 12      | LM       | Loopback Mode.<br>The EMAC_MACCFG.LM bit, when set, direct the MAC to operate in internal loopback mode. (The media independent interface pins are not driven or sampled.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 11      | DM       | Duplex Mode.<br>The EMAC_MACCFG.DM bit, when set, direct the MAC to operate in a Full-Duplex mode where it can transmit and receive simultaneously.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 10      | IPC      | IP Checksum.<br>The EMAC_MACCFG.IPC bit, when set, directs the MAC to calculate the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 25-26 or 29-30 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. This bit, when set, enables IPv4 checksum checking for received frame payloads' TCP/UDP/ICMP headers. When this bit is reset, the Checksum Offload Engine function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared. |



**Table 23-40:** EMAC\_MACCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                 |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 9       | DR       | Disable Retry.<br>The EMAC_MACCFG.DR bit, when set, directs the MAC to attempt only 1 transmission. When a collision occurs on the media independent interface, the MAC will ignore the current frame transmission and report a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC will attempt retries based on the settings of BL. This bit is applicable only to Half-Duplex mode.                                                                                                |                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Retry enabled   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | Retry disabled  |
| 7       | ACS      | Automatic Pad/CRC Stripping.<br>The EMAC_MACCFG.ACS bit, when set, directs the MAC to strip the Pad/FCS field on incoming frames only if the length's field value is less than or equal to 1,500 bytes. All received frames with length field greater than or equal to 1,501 bytes are passed to the application without stripping the Pad/FCS field. When this bit is reset, the MAC will pass all incoming frames to the Host unmodified.                                                                                                |                 |
| 6:5     | BL       | Back Off Limit.<br>The EMAC_MACCFG.BL bit selects the back-off limit, determining the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps and 512 bit times for 10/100 Mbps) the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only to Half-Duplex mode. The random integer r takes the value in the range:<br>0 less-than-equal-to r less-than 2 <sup>k</sup><br>Where k is the minimum of n (number of transmission attempts) or a limit value. |                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | k = min (n, 10) |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | k = min (n, 8)  |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | k = min (n, 4)  |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | k = min (n, 1)  |

**Table 23-40:** EMAC\_MACCFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DC       | <p>Deferral Check.</p> <p>The <code>EMAC_MACCFG.DC</code> bit, when set, enables the deferral check function in the MAC. The MAC will issue a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status when the transmit state machine is deferred for more than 24,288 bit times in 10/100-Mbps mode. If the Jumbo frame mode is enabled in 10/100-Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but is prevented because of an active CRS (carrier sense) signal. Deferral time is not cumulative. If the transmitter defers for 10,000 bit times, then transmits, collides, backs off, and then has to defer again after completion of back-off, the deferral timer resets to 0 and restarts. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in Half-Duplex mode.</p> |
| 3       | TE       | <p>Transmitter Enable.</p> <p>The <code>EMAC_MACCFG.TE</code> bit, when set, enables the transmit state machine of the MAC for transmission. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and will not transmit any further frames.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 2       | RE       | <p>Receiver Enable.</p> <p>The <code>EMAC_MACCFG.RE</code> bit, when set, enables the receiver state machine of the MAC for receiving frames. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and will not receive any further frames.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

## MAC Rx Frame Filter Register

The `EMAC_MACFRMFILT` register controls receive frame filter features.

**EMAC\_MACFRMFILT: MAC Rx Frame Filter Register - R/W**

Reset = 0x0000 0000

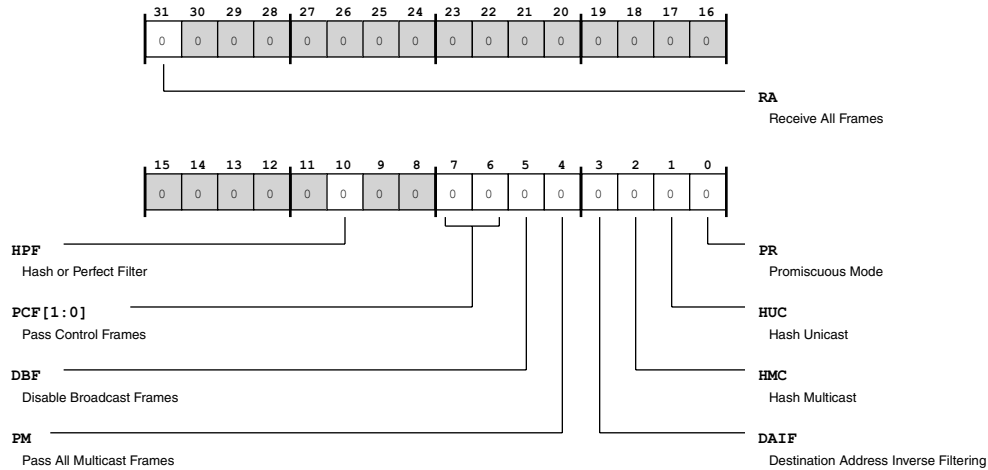


**Figure 23-17:** EMAC\_MACFRMFILT Register Diagram

**Table 23-41:** EMAC\_MACFRMFILT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | RA       | Receive All Frames.<br>The EMAC_MACFRMFILT . RA bit, when set, directs the MAC Receiver module to pass to the Application all frames received irrespective of whether they pass the address filter. The result of the DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes to the Application only those frames that pass the DA address filter. |
| 10      | HPF      | Hash or Perfect Filter.<br>The EMAC_MACFRMFILT . HPF bit. when set, configures the address filter to pass a frame if it matches either the perfect filtering or the hash filtering as set by HMC or HUC bits. When low and if the HUC/HMC bit is set, the frame is passed only if it matches the Hash filter.                                                                                                                               |

**Table 23-41:** EMAC\_MACFRMFILT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                    |                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| 7:6     | PCF      | Pass Control Frames.<br>The EMAC_MACFRMFILT . PCF bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). Note that the processing of PAUSE control frames depends only on RFE of Flow Control Register[2].                      |                                                                                                                                          |
|         |          | 0                                                                                                                                                                                                                                                                          | Pass no control frames<br>All control frames are filtered from reaching the application.                                                 |
|         |          | 1                                                                                                                                                                                                                                                                          | Pass no PAUSE frames<br>All control frames are passed to the application (even if the fail the address filter), except for PAUSE frames. |
|         |          | 2                                                                                                                                                                                                                                                                          | Pass all control frames<br>All control frames are passed to the application (even if the fail the address filter).                       |
|         |          | 3                                                                                                                                                                                                                                                                          | Pass address filtered control frames<br>All control frames that pass the address filter are passed to the application.                   |
| 5       | DBF      | Disable Broadcast Frames.<br>The EMAC_MACFRMFILT . DBF bit, when set, directs the AFM module to filter all incoming broadcast frames. When this bit is reset, the AFM module passes all received broadcast frames                                                          |                                                                                                                                          |
|         |          | 0                                                                                                                                                                                                                                                                          | AFM module passes all received broadcast frames                                                                                          |
|         |          | 1                                                                                                                                                                                                                                                                          | AFM module filters all incoming broadcast frames                                                                                         |
| 4       | PM       | Pass All Multicast Frames.<br>The EMAC_MACFRMFILT . PM bit, when set, indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit. |                                                                                                                                          |

**Table 23-41:** EMAC\_MACFRMFILT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | DAIF     | Destination Address Inverse Filtering.<br>The EMAC_MACFRMFILT.DAIF bit, when set, directs the Address Check block to operate in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.                                                                               |
| 2       | HMC      | Hash Multicast.<br>The EMAC_MACFRMFILT.HMC bit, when set, directs the EMAC to perform destination address filtering of received multicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the DA field with the values programmed in MAC Address Register. |
| 1       | HUC      | Hash Unicast.<br>The EMAC_MACFRMFILT.HUC bit, when set, directs the EMAC to perform destination address filtering of unicast frames according to the hash table. When reset, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the DA field with the values programmed in DA registers.                        |
| 0       | PR       | Promiscuous Mode.<br>The EMAC_MACFRMFILT.PR bit, when set, directs the Address Filter module to pass all incoming frames regardless of its destination or source address. The DA Filter Fails status bits of the Receive Status Word will always be cleared when PR is set.                                                                                    |

## Hash Table High Register

The EMAC\_HASHTBL\_HI register contains the upper 32 bits of the hash table.

EMAC\_HASHTBL\_HI: Hash Table High Register - R/W

Reset = 0x0000 0000

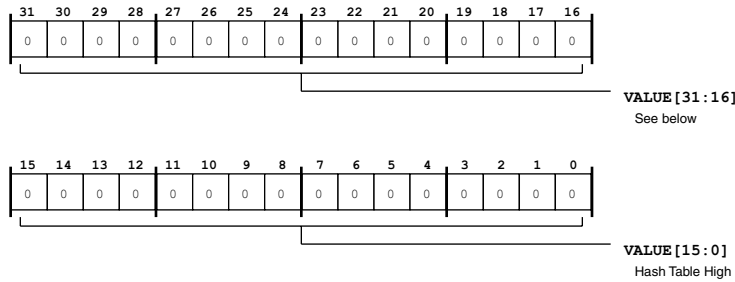


Figure 23-18: EMAC\_HASHTBL\_HI Register Diagram

Table 23-42: EMAC\_HASHTBL\_HI Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Hash Table High.<br>The EMAC_HASHTBL_HI.VALUE bits contain the upper 32 bits of Hash table. |

Hash Table Low Register

The EMAC\_HASHTBL\_LO register contains the lower 32 bits of the hash table.

EMAC\_HASHTBL\_LO: Hash Table Low Register - R/W

Reset = 0x0000 0000

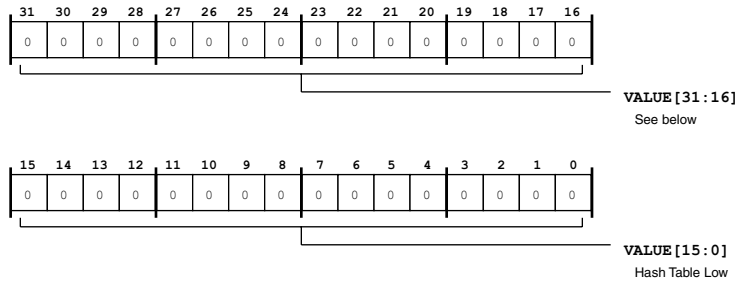


Figure 23-19: EMAC\_HASHTBL\_LO Register Diagram

**Table 23-43:** EMAC\_HASHTBL\_LO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Hash Table Low.<br>The EMAC_HASHTBL_LO.VALUE bits contain the lower 32 bits of Hash table. |

## SMI Address Register

The EMAC\_SMI\_ADDR register contains the station management interface address and feature settings.

**EMAC\_SMI\_ADDR: SMI Address Register - R/W**

Reset = 0x0000 0000

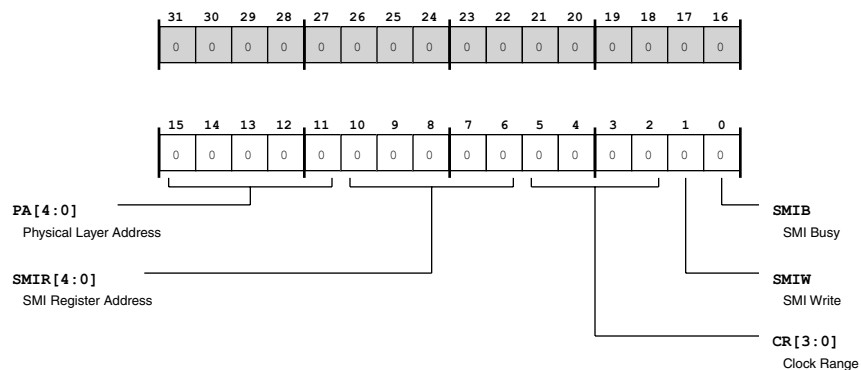


**Figure 23-20:** EMAC\_SMI\_ADDR Register Diagram

**Table 23-44:** EMAC\_SMI\_ADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:11   | PA       | Physical Layer Address.<br>The EMAC_SMI_ADDR.PA bits select the PHY. This field tells which of the 32 possible PHY devices are being accessed. |
| 10:6    | SMIR     | SMI Register Address.<br>The EMAC_SMI_ADDR.SMIR bits select the desired Station Management Interface register in the selected PHY device.      |

**Table 23-44:** EMAC\_SMI\_ADDR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:2     | CR       | <p><b>Clock Range.</b><br/>The EMAC_SMI_ADDR.CR bits select the Clock Range, determining the frequency of the MDC clock as per the "SCLK" frequency. The suggested range of SCLK frequency applicable for each value below (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz. When bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when SCLK is of frequency 100 MHz and you program these bits as "1010", then the resultant MDC clock will be of 12.5 MHz which is outside the limit of IEEE 802.3 specified range. Please program the values given below only if the interfacing chips supports faster MDC clocks.</p> |
|         |          | 0 MDC Clock=SCLK/42<br>(for SCLK=60-100MHz)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 MDC Clock= SCLK/62<br>(for SCLK=100-125 MHz)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 2 MDC Clock= SCLK/16<br>(for SCLK=20-35 MHz)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 3 MDC Clock= SCLK/26<br>(for SCLK=35-60 MHz)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 8 MDC Clock=SCLK/4                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 9 MDC Clock=SCLK/6                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 10 MDC Clock=SCLK/8                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 11 MDC Clock=SCLK/10                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 12 MDC Clock=SCLK/12                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 13 MDC Clock=SCLK/14                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 14 MDC Clock=SCLK/16                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 15 MDC Clock=SCLK/18                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |



**Table 23-44:** EMAC\_SMI\_ADDR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | SMIW     | SMI Write.<br>The EMAC_SMI_ADDR.SMIW bit, when set, tells the PHY that this will be a Write operation using the Station Management Interface Data register. If this bit is not set, this will be a Read operation.                                                                                                                                                                                                                                                                                                                                              |
| 0       | SMIB     | SMI Busy.<br>The EMAC_SMI_ADDR.SMIB bit should read a logic 0 before writing to SMI_ADDR and SMI_DATA. This bit must also be set to 0 during a Write to SMI_ADDR. During a PHY register access, this bit will be set to 1'b1 by the Application to indicate that a Read or Write access is in progress. SMI_DATA should be kept valid until this bit is cleared by the MAC during a PHY Write operation. SMI_DATA is invalid until this bit is cleared by the MAC during a PHY Read operation. The SMI_ADDR should not be written to until this bit is cleared. |

## SMI Data Register

The EMAC\_SMI\_DATA register contains the station management interface data.

### EMAC\_SMI\_DATA: SMI Data Register - R/W

Reset = 0x0000 0000

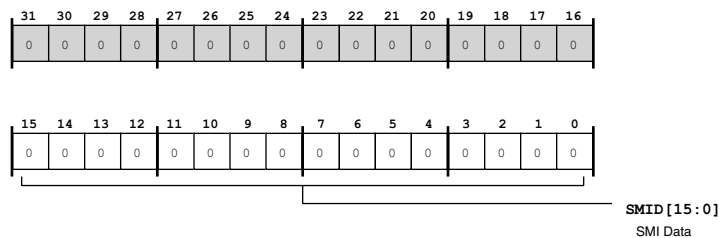


**Figure 23-21:** EMAC\_SMI\_DATA Register Diagram

**Table 23-45:** EMAC\_SMI\_DATA Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | SMID     | SMI Data.<br>The EMAC_SMI_DATA.SMID bits contain the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation. |

## Flow Control Register

The EMAC\_FLOWCTL register controls EMAC flow control features.

**EMAC\_FLOWCTL: Flow Control Register - R/W**

Reset = 0x0000 0000

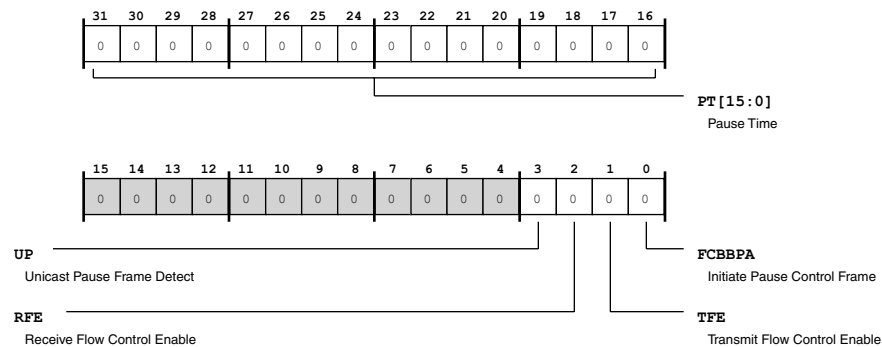


Figure 23-22: EMAC\_FLOWCTL Register Diagram

Table 23-46: EMAC\_FLOWCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:16   | PT       | Pause Time.<br>The EMAC_FLOWCTL . PT bits hold the value to be used in the Pause Time field in the transmit control frame.                                                                                                                                                                                                                                                                                                               |
| 3       | UP       | Unicast Pause Frame Detect.<br>The EMAC_FLOWCTL . UP bit, when set, directs the MAC to detect the Pause frames with the station's unicast address specified in MAC Address0 High Register and MAC Address0 Low Register, in addition to the detecting Pause frames with the unique multicast address. When this bit is reset, the MAC will detect only a Pause frame with the unique multicast address specified in the 802.3x standard. |
| 2       | RFE      | Receive Flow Control Enable.<br>The EMAC_FLOWCTL . RFE bit, when set, directs the MAC to decode the received Pause frame and disable its transmitter for a specified (Pause Time) time. When this bit is reset, the decode function of the Pause frame is disabled.                                                                                                                                                                      |

**Table 23-46:** EMAC\_FLOWCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | TFE      | Transmit Flow Control Enable.<br>In Full-Duplex mode, when the EMAC_FLOWCTL.TFE bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC will not transmit any Pause frames. In Half-Duplex mode, when this bit is set, the MAC enables the back pressure operation. When this bit is reset, the back pressure feature is disabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 0       | FCBBPA   | Initiate Pause Control Frame.<br>The EMAC_FLOWCTL.FCBBPA bit initiates a Pause Control frame in Full-Duplex mode and activates the backpressure function in Half-Duplex mode if TFE bit is set. In Full-Duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit will continue to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC will reset this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared. In Half-Duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC Core. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically OR'ed with the flow control input signal for the backpressure function. When the MAC is configured to Full-Duplex mode, the BPA is automatically disabled. |

## VLAN Tag Register

The EMAC\_VLANTAG register contains the VLAN tag.

EMAC\_VLANTAG: VLAN Tag Register - R/W

Reset = 0x0000 0000

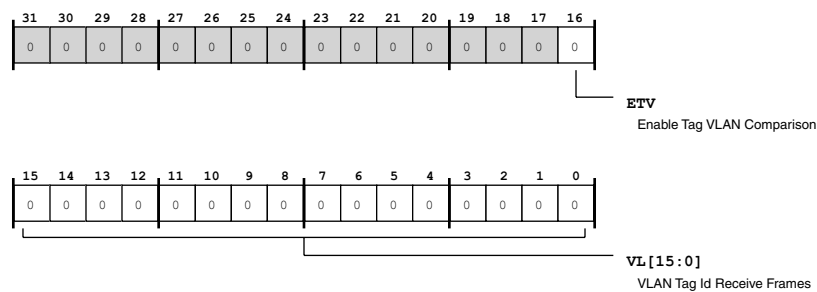


Figure 23-23: EMAC\_VLANTAG Register Diagram

Table 23-47: EMAC\_VLANTAG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | ETV      | Enable Tag VLAN Comparison.<br>The EMAC_VLANTAG.ETV bit, when set, directs the EMAC to use a 12-bit VLAN identifier, rather than the complete 16-bit VLAN tag, for comparison and filtering. Bits[11:0] of the VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. When this bit is reset, all 16 bits of the received VLAN frame's fifteenth and sixteenth bytes are used for comparison.                                                                                                                                                                                                                             |
| 15:0    | VL       | VLAN Tag Id Receive Frames.<br>The EMAC_VLANTAG.VL bits contain the 802.1Q VLAN tag to identify VLAN frames, and is compared to the fifteenth and sixteenth bytes of the frames being received for VLAN frames. Bits[15:13] are the User Priority, Bit[12] is the Canonical Format Indicator (CFI) and bits[11:0] are the VLAN tag's VLAN Identifier (VID) field. When the ETV bit is set, only the VID (Bits[11:0]) is used for comparison. If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and sixteenth bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 to be VLAN frames. |

Debug Register

The EMAC\_DBG register contains EMAC debug status information.

**EMAC\_DBG: Debug Register - R/W**

Reset = 0x0000 0000

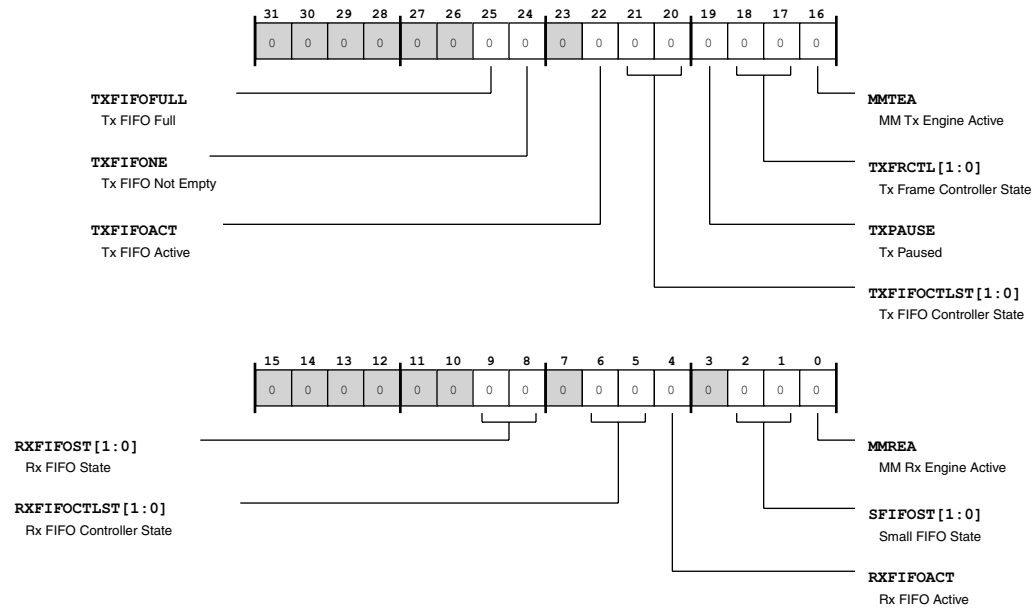


**Figure 23-24:** EMAC\_DBG Register Diagram

**Table 23-48:** EMAC\_DBG Register Fields

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                              |
|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | TXFIFOFULL  | Tx FIFO Full.<br>The EMAC_DBG.TXFIFOFULL bit, when high, indicates that the MFL TxStatus FIFO is full and hence the MFL cannot accept any more frames for transmission.                                                                                                              |
| 24      | TXFIFONE    | Tx FIFO Not Empty.<br>The EMAC_DBG.TXFIFONE bit, when high, indicates that the MFL TxFIFO is not empty and has some data left for transmission.                                                                                                                                      |
| 22      | TXFIFOACT   | Tx FIFO Active.<br>The EMAC_DBG.TXFIFOACT bit, when high, indicates that the MFL TxFIFO write controller is active and transferring data to the TxFIFO.                                                                                                                              |
| 21:20   | TXFIFOCTLST | Tx FIFO Controller State.<br>The EMAC_DBG.TXFIFOCTLST bits indicate the state of the TxFIFO read controller: 00 IDLE state 01 READ state (transferring data to MAC transmitter) 10 Waiting for TxStatus from MAC transmitter 11 Writing the received TxStatus or flushing the TxFIFO |

**Table 23-48:** EMAC\_DBG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19      | TXPAUSE  | Tx Paused.<br>The EMAC_DBG.TXPAUSE bit, when high, indicates that the MAC transmitter is in PAUSE condition (in full-duplex only) and hence does not schedule any frame for transmission. |
| 18:17   | TXFRCTL  | Tx Frame Controller State.<br>The EMAC_DBG.TXFRCTL bits indicate the state of the MAC transmit frame controller module.                                                                   |
|         |          | 0<br>Idle<br>Frame controller is in idle state.                                                                                                                                           |
|         |          | 1<br>Wait<br>Frame controller is waiting for status of previous frame or IFG/backoff period end.                                                                                          |
|         |          | 2<br>Pause<br>Frame controller is generating and transmitting a PAUSE control frame (in full duplex mode).                                                                                |
|         |          | 3<br>Transmit<br>Frame controller is transferring input frame for transmission.                                                                                                           |
| 16      | MMTEA    | MM Tx Engine Active.<br>The EMAC_DBG.MMTEA bit, when high, indicates that the MAC core transmit protocol engine is actively transmitting data and not in IDLE state.                      |
| 9:8     | RXFIFOST | Rx FIFO State.<br>The EMAC_DBG.RXFIFOST bits give the status of the RxFIFO fill level and relationship to the flow-control activation threshold.                                          |
|         |          | 0<br>Rx FIFO Empty                                                                                                                                                                        |
|         |          | 1<br>Rx FIFO Below De-activate FCT                                                                                                                                                        |
|         |          | 2<br>Rx FIFO Above De-activate FCT                                                                                                                                                        |
|         |          | 3<br>Rx FIFO Full                                                                                                                                                                         |

**Table 23-48:** EMAC\_DBG Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                    |
|---------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:5     | RXFIFOCTLST | Rx FIFO Controller State.<br>The EMAC_DBG.RXFIFOCTLST bits give the state of the RxFIFO read controller.                                                                                   |
|         |             | 0<br>Idle<br>Read controller is in idle state.                                                                                                                                             |
|         |             | 1<br>Read Data<br>Read controller is reading frame data.                                                                                                                                   |
|         |             | 2<br>Read Status<br>Read controller is reading frame status or time-stamp.                                                                                                                 |
|         |             | 3<br>Flush<br>Read controller is flushing the frame data and status.                                                                                                                       |
| 4       | RXFIFOACT   | Rx FIFO Active.<br>The EMAC_DBG.RXFIFOACT bit, when high, indicates that the MFL RxFIFO write controller is active and transferring a received frame to the FIFO.                          |
| 2:1     | SFIFOST     | Small FIFO State.<br>The EMAC_DBG.SFIFOST bit, when high, indicates the active state of the small FIFO read and write controllers respectively of the MAC receive frame controller module. |
| 0       | MMREA       | MM Rx Engine Active.<br>The EMAC_DBG.MMREA bit, when high, indicates that the MAC core receive protocol engine is actively receiving data and not in IDLE state.                           |

## Interrupt Status Register

The EMAC\_ISTAT register indicates EMAC interrupt status.

**EMAC\_ISTAT: Interrupt Status Register - R/W**

Reset = 0x0000 0000

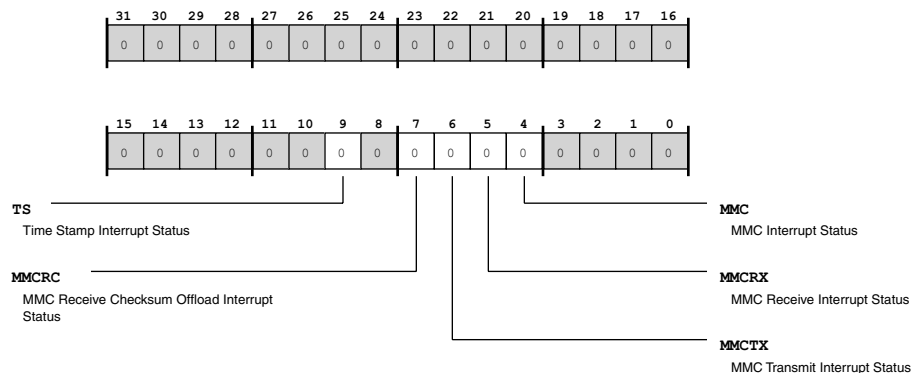


**Figure 23-25:** EMAC\_ISTAT Register Diagram

**Table 23-49:** EMAC\_ISTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | TS       | <p>Time Stamp Interrupt Status.</p> <p>When Advanced Time Stamping feature is enabled, the EMAC_ISTAT.TS bit is set when:</p> <ul style="list-style-type: none"> <li>The system time value equals or exceeds the value specified in the Target Time High and Low registers, or</li> <li>There is an overflow in the seconds register, or</li> <li>When the Auxiliary snapshot trigger is asserted.</li> </ul> <p>This bit is cleared on reading the byte 0 of the "Time Stamp Status register". Otherwise, when default Time-Stamping is enabled, this bit when set indicates that the system time value equals or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this Interrupt Status Register. In all other modes, this bit is reserved.</p> |
| 7       | MMCRC    | <p>MMC Receive Checksum Offload Interrupt Status.</p> <p>The EMAC_ISTAT.MMCRC bit is set high whenever an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 6       | MMCTX    | <p>MMC Transmit Interrupt Status.</p> <p>The EMAC_ISTAT.MMCTX bit is set high whenever an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |



**Table 23-49:** EMAC\_ISTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | MMCRX    | MMC Receive Interrupt Status.<br>The EMAC_ISTAT.MMCRX bit is set high whenever an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all the bits in this interrupt register are cleared. |
| 4       | MMC      | MMC Interrupt Status.<br>The EMAC_ISTAT.MMC bit is set high whenever any of bits 7:5 is set high and cleared only when all of these bits are low.                                                                             |

## Interrupt Mask Register

The EMAC\_IMSK register enables (unmasks) EMAC interrupts.

**EMAC\_IMSK: Interrupt Mask Register - R/W**

Reset = 0x0000 0000

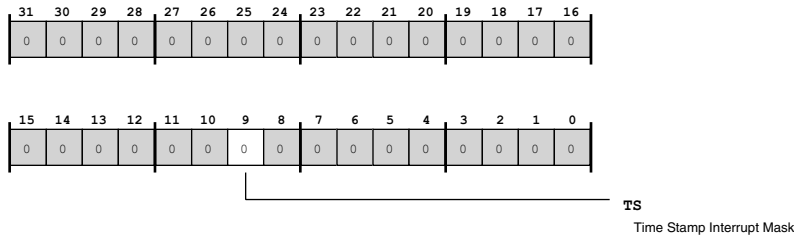


**Figure 23-26:** EMAC\_IMSK Register Diagram

**Table 23-50:** EMAC\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | TS       | Time Stamp Interrupt Mask.<br>The EMAC_IMSK.TS bit, when set, disables the assertion of the interrupt signal due to the setting of Time Stamp Interrupt Status bit in ISTAT Register. |

## MAC Address 0 High Register

The EMAC\_ADDRO\_HI register holds the address 0 high bits.

EMAC\_ADDR0\_HI: MAC Address 0 High Register - R/W

Reset = 0x8000 ffff

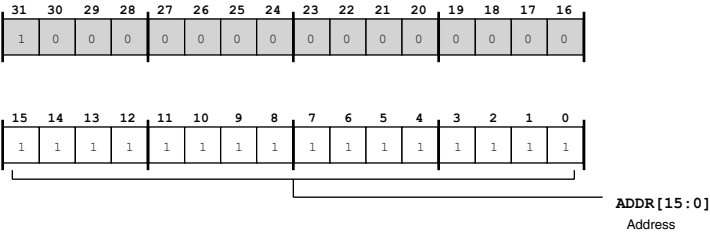


Figure 23-27: EMAC\_ADDR0\_HI Register Diagram

Table 23-51: EMAC\_ADDR0\_HI Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | ADDR     | Address.<br>The EMAC_ADDR0_HI . ADDR bits contain the upper 16 bits (47:32) of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames |

MAC Address 0 Low Register

The EMAC\_ADDR0\_LO register holds the address 0 low bits.

EMAC\_ADDR0\_LO: MAC Address 0 Low Register - R/W

Reset = 0xffff ffff

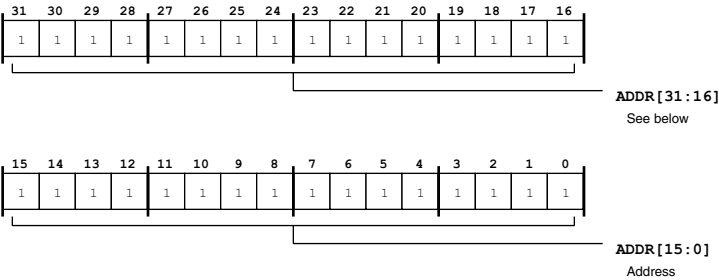


Figure 23-28: EMAC\_ADDR0\_LO Register Diagram

Table 23-52: EMAC\_ADDR0\_LO Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | ADDR     | Address.<br>The EMAC_ADDR0_LO.ADDR bits contain the lower 32 bits of the 6-byte first MAC address. This is used by the MAC for filtering for received frames and for inserting the MAC address in the Transmit Flow Control (PAUSE) Frames. |

MMC Control Register

The EMAC\_MMC\_CTL register selects the MMC operating mode.

EMAC\_MMC\_CTL: MMC Control Register - R/W

Reset = 0x0000 0000

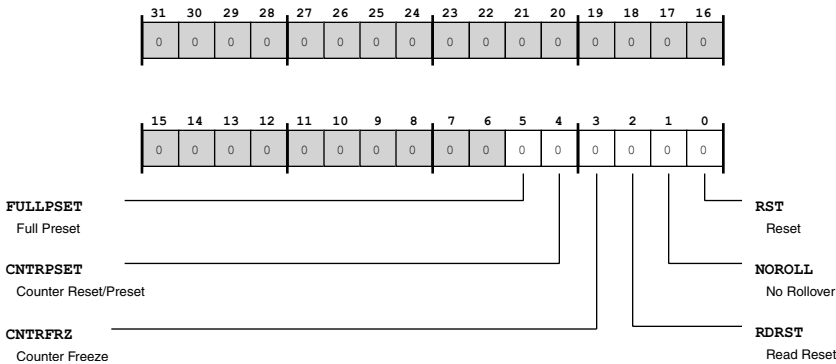


Figure 23-29: EMAC\_MMC\_CTL Register Diagram

**Table 23-53:** EMAC\_MMC\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5       | FULLPSET | <p>Full Preset.</p> <p>The EMAC_MMC_CTL.FULLPSET bit, when low and bit 4 is set, presets all MMC counters to almost-half value. All octet counters get preset to 0x7FFF_F800 (half - 2KBytes) and all frame-counters gets preset to 0x7FFF_FFF0 (half - 16). When high and bit4 is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (full - 2KBytes) and all frame-counters gets preset to 0xFFFF_FFF0 (full - 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and frame counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.</p> |
| 4       | CNTRPSET | <p>Counter Reset/Preset.</p> <p>The EMAC_MMC_CTL.CNTRPSET bit, when set, initializes all counters or presets counters to almost full or almost half as per Bit 5. This bit is cleared automatically after 1 clock cycle. This bit along with bit5 is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.</p>                                                                                                                                                                                                                                                                                                                 |
| 3       | CNTRFRZ  | <p>Counter Freeze.</p> <p>The EMAC_MMC_CTL.CNTRFRZ bit, when set, freezes all the MMC counters to their current value. (None of the MMC counters are updated because of any transmitted or received frame until this bit is reset to 0. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.)</p>                                                                                                                                                                                                                                                                                                                                  |
| 2       | RDRST    | <p>Read Reset.</p> <p>The EMAC_MMC_CTL.RDRST bit, when set, resets the MMC counters to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (bits[7:0]) is read.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 1       | NOROLL   | <p>No Rollover.</p> <p>The EMAC_MMC_CTL.NOROLL bit, when set, prevents counter rolls over to 0 after reaching max.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0       | RST      | <p>Reset.</p> <p>The EMAC_MMC_CTL.RST bit, when set, resets all counters. This bit is cleared automatically after 1 clock cycle</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## MMC Rx Interrupt Register

The EMAC\_MMC\_RXINT register indicates status of MMC receive interrupts.

### EMAC\_MMC\_RXINT: MMC Rx Interrupt Register - R/W

Reset = 0x0000 0000

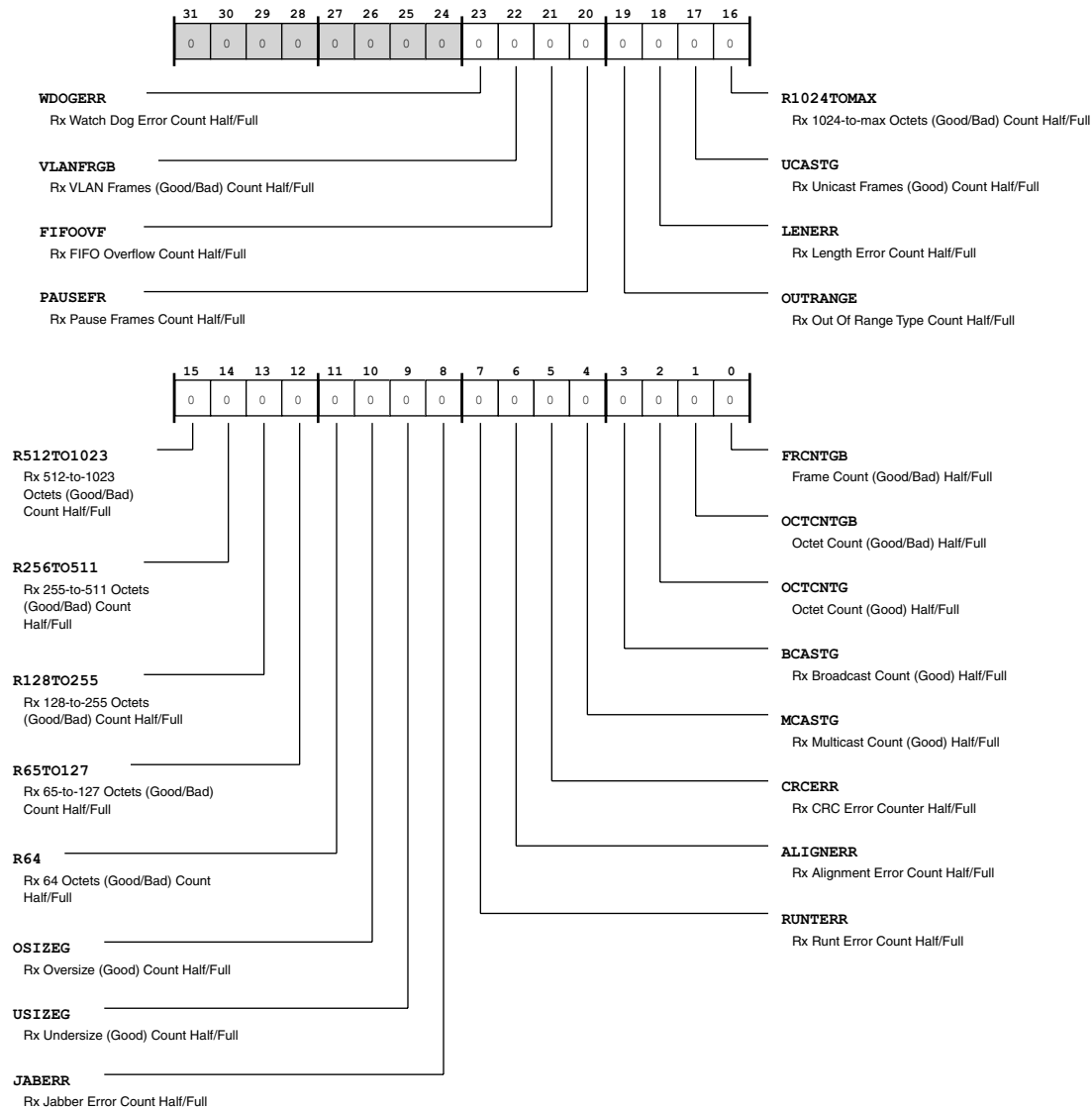


Figure 23-30: EMAC\_MMC\_RXINT Register Diagram

**Table 23-54:** EMAC\_MMC\_RXINT Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                               |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23      | WDOGERR    | Rx Watch Dog Error Count Half/Full.<br>The EMAC_MMC_RXINT.WDOGERR bit is set when rxwatchdog_error counter reaches full or half.                      |
| 22      | VLANFRGB   | Rx VLAN Frames (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.VLANFRGB bit is set when rxvlanframes_gb counter reaches full or half.               |
| 21      | FIFOOVF    | Rx FIFO Overflow Count Half/Full.<br>The EMAC_MMC_RXINT.FIFOOVF bit is set when rxffiooverflow counter reaches full or half.                          |
| 20      | PAUSEFR    | Rx Pause Frames Count Half/Full.<br>The EMAC_MMC_RXINT.PAUSEFR bit is set when rxpauseframes counter reaches full or half.                            |
| 19      | OUTRANGE   | Rx Out Of Range Type Count Half/Full.<br>The EMAC_MMC_RXINT.OUTRANGE bit is set when rxoutofrangetype counter reaches full or half.                   |
| 18      | LENERR     | Rx Length Error Count Half/Full.<br>The EMAC_MMC_RXINT.LENERR bit is set when rxlengtherror counter reaches full or half.                             |
| 17      | UCASTG     | Rx Unicast Frames (Good) Count Half/Full.<br>The EMAC_MMC_RXINT.UCASTG bit is set when rxunicastframes_g counter reaches full or half.                |
| 16      | R1024TOMAX | Rx 1024-to-max Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R1024TOMAX bit is set when rx1024tomaxoctets_gb counter reaches full or half. |
| 15      | R512TO1023 | Rx 512-to-1023 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R512TO1023 bit is set when rx512to1023octets_gb counter reaches full or half. |
| 14      | R256TO511  | Rx 255-to-511 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R256TO511 bit is set when rx255to511octets_gb counter reaches full or half.    |

**Table 23-54:** EMAC\_MMC\_RXINT Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                            |
|---------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | R128TO255 | Rx 128-to-255 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R128TO255 bit is set when rx128to255octets_gb counter reaches full or half. |
| 12      | R65TO127  | Rx 65-to-127 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R65TO127 bit is set when rx65to127octets_gb counter reaches full or half.    |
| 11      | R64       | Rx 64 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_RXINT.R64 bit is set when rx64octets_gb counter reaches full or half.                     |
| 10      | OSIZEG    | Rx Oversize (Good) Count Half/Full.<br>The EMAC_MMC_RXINT.OSIZEG bit is set when rxoversize_g counter reaches full or half.                        |
| 9       | USIZEG    | Rx Undersize (Good) Count Half/Full.<br>The EMAC_MMC_RXINT.USIZEG bit is set when rxundersize_g counter reaches full or half.                      |
| 8       | JABERR    | Rx Jabber Error Count Half/Full.<br>The EMAC_MMC_RXINT.JABERR bit is set when rxjabbererror counter reaches full or half.                          |
| 7       | RUNTERR   | Rx Runt Error Count Half/Full.<br>The EMAC_MMC_RXINT.RUNTERR bit is set when rxrunterror counter reaches full or half.                             |
| 6       | ALIGNERR  | Rx Alignment Error Count Half/Full.<br>The EMAC_MMC_RXINT.ALIGNERR bit is set when rxalignmenterror counter reaches full or half                   |
| 5       | CRCERR    | Rx CRC Error Counter Half/Full.<br>The EMAC_MMC_RXINT.CRCERR bit is set when rxcrcerror counter reaches full or half.                              |
| 4       | MCASTG    | Rx Multicast Count (Good) Half/Full.<br>The EMAC_MMC_RXINT.MCASTG bit is set when rxmulticastframes_g counter reaches full or half.                |
| 3       | BCASTG    | Rx Broadcast Count (Good) Half/Full.<br>The EMAC_MMC_RXINT.BCASTG bit is set when rxbroadcastframes_g counter reaches full or half.                |

**Table 23-54:** EMAC\_MMC\_RXINT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 2       | OCTCNTG  | Octet Count (Good) Half/Full.<br>The EMAC_MMC_RXINT.OCTCNTG bit is set when rxoctetcount_g counter reaches full or half.       |
| 1       | OCTCNTGB | Octet Count (Good/Bad) Half/Full.<br>The EMAC_MMC_RXINT.OCTCNTGB bit is set when rxoctetcount_gb counter reaches half or full. |
| 0       | FRCNTGB  | Frame Count (Good/Bad) Half/Full.<br>The EMAC_MMC_RXINT.FRCNTGB bit is set when rxframecount_gb counter reaches half or full.  |

## MMC Tx Interrupt Register

The EMAC\_MMC\_TXINT register indicates status of MMC transmit interrupts.



# EMAC\_MMC\_TXINT: MMC Tx Interrupt Register - R/W

Reset = 0x0000 0000

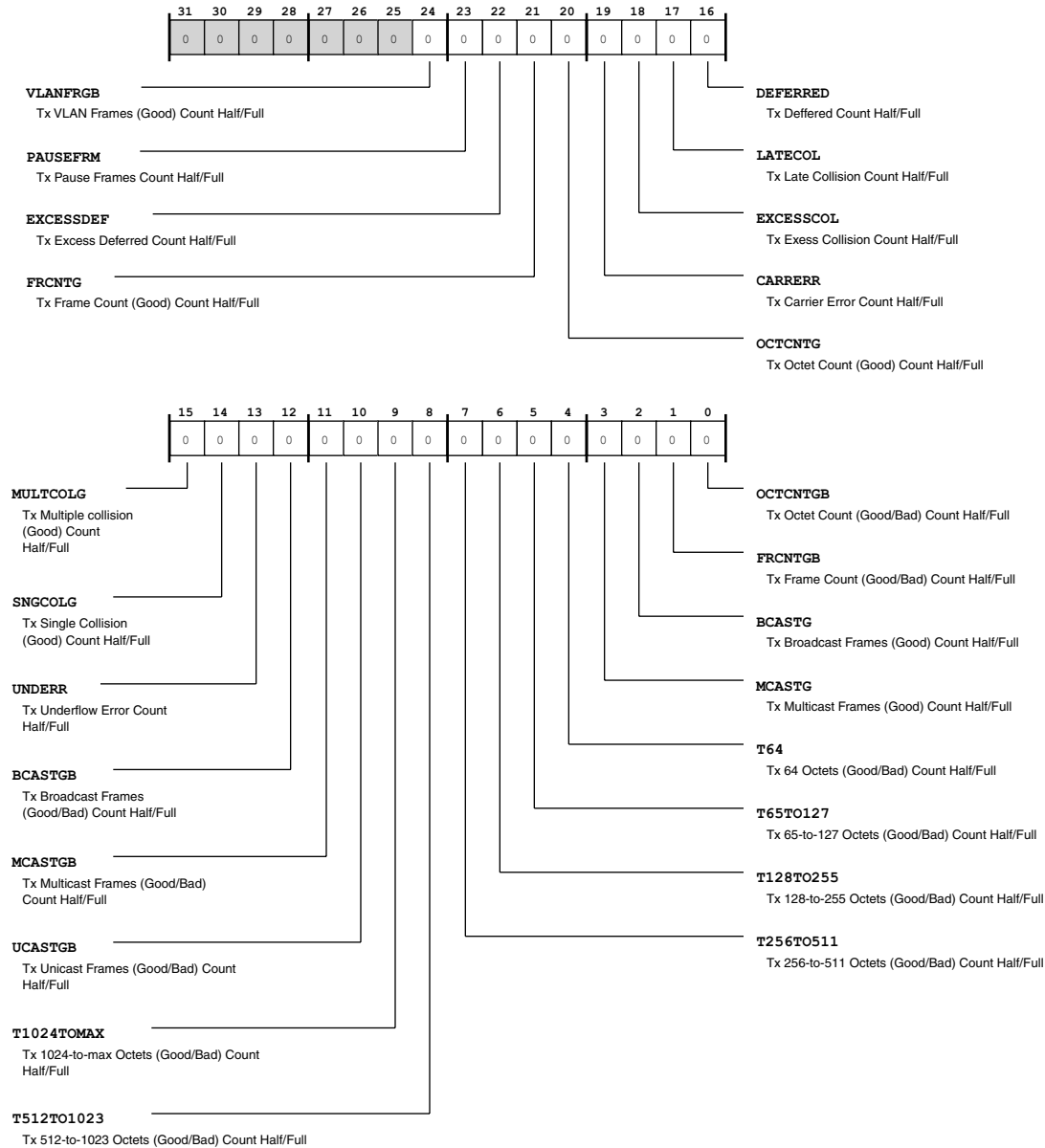


Figure 23-31: EMAC\_MMC\_TXINT Register Diagram

**Table 23-55:** EMAC\_MMC\_TXINT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                 |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 24      | VLANFRGB  | Tx VLAN Frames (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.VLANFRGB bit is set when txvlanframes_g counter reaches full or half.      |
| 23      | PAUSEFRM  | Tx Pause Frames Count Half/Full.<br>The EMAC_MMC_TXINT.PAUSEFRM bit is set when txpauseframes counter reaches full or half.             |
| 22      | EXCESSDEF | Tx Excess Deferred Count Half/Full.<br>The EMAC_MMC_TXINT.EXCESSDEF bit is set when txexcessdef counter reaches full or half.           |
| 21      | FRCNTG    | Tx Frame Count (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.FRCNTG bit is set when txframecount_g counter reaches full or half.        |
| 20      | OCTCNTG   | Tx Octet Count (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.OCTCNTG bit is set when txoctetcount_g counter reaches full or half.       |
| 19      | CARRERR   | Tx Carrier Error Count Half/Full.<br>The EMAC_MMC_TXINT.CARRERR bit is set when txcarriererr counter reaches full or half.              |
| 18      | EXCESSCOL | Tx Excess Collision Count Half/Full.<br>The EMAC_MMC_TXINT.EXCESSCOL bit is set when txexesscol counter reaches full or half.           |
| 17      | LATECOL   | Tx Late Collision Count Half/Full.<br>The EMAC_MMC_TXINT.LATECOL bit is set when txlatecol counter reaches full or half.                |
| 16      | DEFERRED  | Tx Deffered Count Half/Full.<br>The EMAC_MMC_TXINT.DEFERRED bit is set when txdeffered counter reaches full or half.                    |
| 15      | MULTCOLG  | Tx Multiple collision (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.MULTCOLG bit is set when txmulticol_g counter reaches full or half. |

**Table 23-55:** EMAC\_MMC\_TXINT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                               |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | SNGCOLG    | Tx Single Collision (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.SNGCOLG bit is set when txsinglecol_g counter reaches full or half.                 |
| 13      | UNDERR     | Tx Underflow Error Count Half/Full.<br>The EMAC_MMC_TXINT.UNDERR bit is set when txunderflowerror counter reaches full or half.                       |
| 12      | BCASTGB    | Tx Broadcast Frames (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.BCASTGB bit is set when txbroadcastframes_gb counter reaches full or half.      |
| 11      | MCASTGB    | Tx Multicast Frames (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.MCASTGB bit is set when txmulticastframes_gb counter reaches full or half.      |
| 10      | UCASTGB    | Tx Unicast Frames (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.UCASTGB bit is set when txunicastframes_gb counter reaches full or half.          |
| 9       | T1024TOMAX | Tx 1024-to-max Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T1024TOMAX bit is set when tx1024tomaxoctets_gb counter reaches full or half. |
| 8       | T512TO1023 | Tx 512-to-1023 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T512TO1023 bit is set when tx512to1023octets_gb counter reaches full or half. |
| 7       | T256TO511  | Tx 256-to-511 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T256TO511 bit is set when tx256to511octets_gb counter reaches full or half.    |
| 6       | T128TO255  | Tx 128-to-255 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T128TO255 bit is set when tx128to255octets_gb counter reaches full or half.    |
| 5       | T65TO127   | Tx 65-to-127 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T65TO127 bit is set when tx65to127octets_gb counter reaches full or half.       |
| 4       | T64        | Tx 64 Octets (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.T64 bit is set when tx64octets_gb counter reaches full or half.                        |

**Table 23-55:** EMAC\_MMC\_TXINT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | MCASTG   | Tx Multicast Frames (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.MCASTG bit is set when txmulticastframes_g counter reaches full or half. |
| 2       | BCASTG   | Tx Broadcast Frames (Good) Count Half/Full.<br>The EMAC_MMC_TXINT.BCASTG bit is set when txbroadcastframes_g counter reaches full or half. |
| 1       | FRCNTGB  | Tx Frame Count (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.FRCNTGB bit is set when txframecount_gb counter reaches full or half.     |
| 0       | OCTCNTGB | Tx Octet Count (Good/Bad) Count Half/Full.<br>The EMAC_MMC_TXINT.OCTCNTGB bit is set when txoctetcount_gb counter reaches full or half.    |

## MMC Rx Interrupt Mask Register

The EMAC\_MMC\_RXIMSK register enables (unmasks) MMC receive interrupts.

# EMAC\_MMC\_RXIMSK: MMC Rx Interrupt Mask Register - R/W

Reset = 0x0000 0000

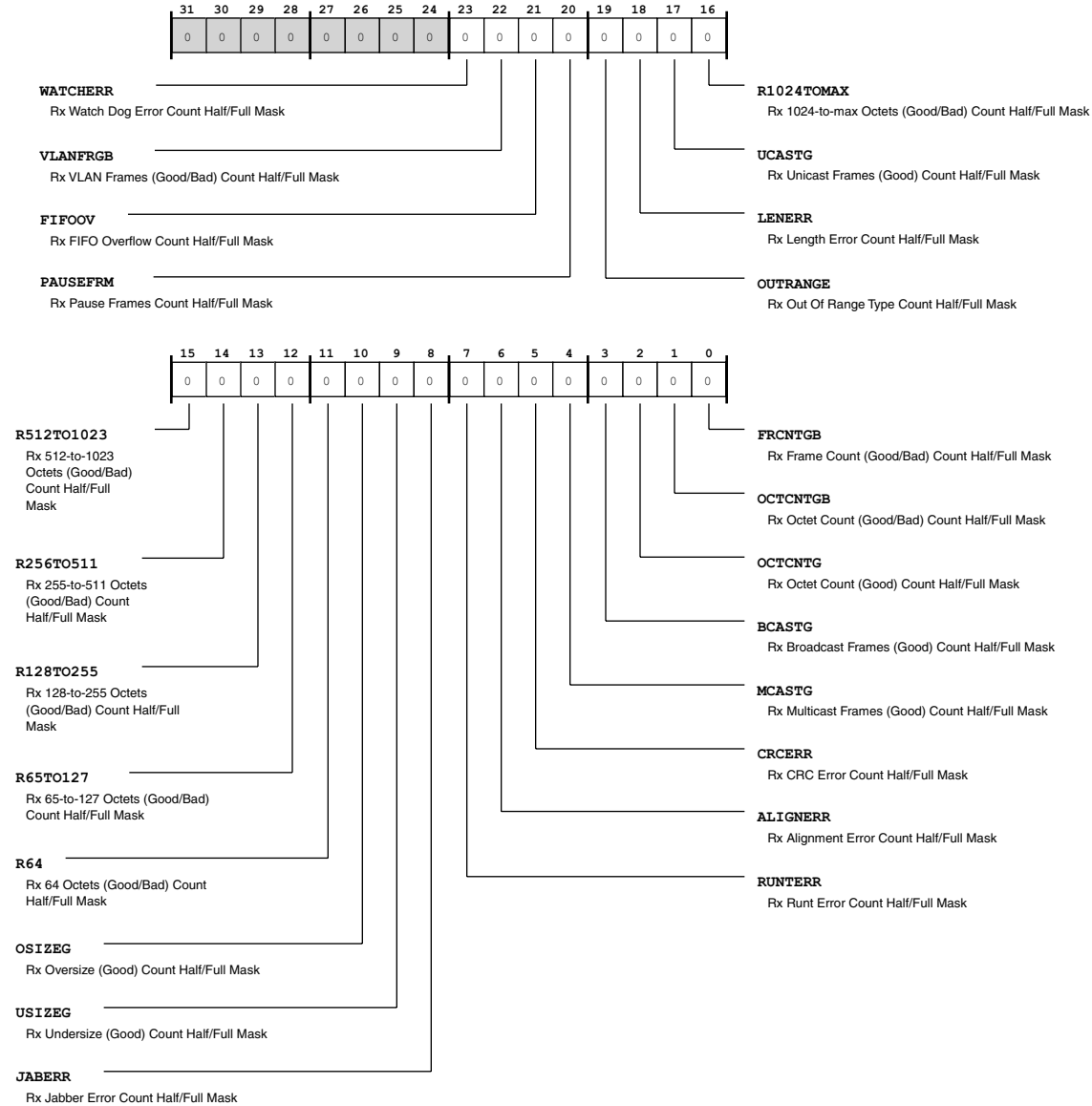


Figure 23-32: EMAC\_MMC\_RXIMSK Register Diagram

Table 23-56: EMAC\_MMC\_RXIMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23      | WATCHERR | Rx Watch Dog Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.WATCHERR bit, when set, masks the interrupt when rxwatchdog_error counter reaches full or half. |

**Table 23-56:** EMAC\_MMC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                             |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | VLANFRGB   | Rx VLAN Frames (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.VLANFRGB bit, when set, masks the interrupt when rxvlanframes_gb counter reaches full or half.               |
| 21      | FIFOOV     | Rx FIFO Overflow Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.FIFOOV bit, when set, masks the interrupt when rxffiooverflow counter reaches full or half.                           |
| 20      | PAUSEFRM   | Rx Pause Frames Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.PAUSEFRM bit, when set, masks the interrupt when rxpauseframes counter reaches full or half.                           |
| 19      | OUTRANGE   | Rx Out Of Range Type Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.OUTRANGE bit, when set, masks the interrupt when rxoutofrangetype counter reaches full or half.                   |
| 18      | LENERR     | Rx Length Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.LENERR bit, when set, masks the interrupt when rxlengtherror counter reaches full or half.                             |
| 17      | UCASTG     | Rx Unicast Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.UCASTG bit, when set, masks the interrupt when rxunicastframes_g counter reaches full or half.                |
| 16      | R1024TOMAX | Rx 1024-to-max Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R1024TOMAX bit, when set, masks the interrupt when rx1024tomaxoctets_gb counter reaches full or half. |
| 15      | R512TO1023 | Rx 512-to-1023 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R512TO1023 bit, when set, masks the interrupt when rx512to1023octets_gb counter reaches full or half. |
| 14      | R256TO511  | Rx 255-to-511 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R256TO511 bit, when set, masks the interrupt when rx255to511octets_gb counter reaches full or half.    |
| 13      | R128TO255  | Rx 128-to-255 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R128TO255 bit, when set, masks the interrupt when rx128to255octets_gb counter reaches full or half.    |
| 12      | R65TO127   | Rx 65-to-127 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R65TO127 bit, when set, masks the interrupt when rx65to127octets_gb counter reaches full or half.       |

**Table 23-56:** EMAC\_MMC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | R64      | Rx 64 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.R64 bit, when set, masks the interrupt when rx64octets_gb counter reaches full or half.             |
| 10      | OSIZEG   | Rx Oversize (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.OSIZEG bit, when set, masks the interrupt when rxoversize_g counter reaches full or half.                |
| 9       | USIZEG   | Rx Undersize (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.USIZEG bit, when set, masks the interrupt when rxundersize_g counter reaches full or half.              |
| 8       | JABERR   | Rx Jabber Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.JABERR bit, when set, masks the interrupt when rxjabbererror counter reaches full or half.                  |
| 7       | RUNTERR  | Rx Runt Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.RUNTERR bit, when set, masks the interrupt when rxrunterror counter reaches full or half.                     |
| 6       | ALIGNERR | Rx Alignment Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.ALIGNERR bit, when set, masks the interrupt when rxalignmenterror counter reaches full or half.          |
| 5       | CRCERR   | Rx CRC Error Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.CRCERR bit, when set, masks the interrupt when rxrcerror counter reaches full or half.                         |
| 4       | MCASTG   | Rx Multicast Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.MCASTG bit, when set, masks the interrupt when rxmulticastframes_g counter reaches full or half. |
| 3       | BCASTG   | Rx Broadcast Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.BCASTG bit, when set, masks the interrupt when rxbroadcastframes_g counter reaches full or half. |
| 2       | OCTCNTG  | Rx Octet Count (Good) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.OCTCNTG bit, when set, masks the interrupt when rxoctetcount_g counter reaches full or half.          |
| 1       | OCTCNTGB | Rx Octet Count (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.OCTCNTGB bit, when set, masks the interrupt when rxoctetcount_gb counter reaches half or full.    |

**Table 23-56:** EMAC\_MMC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | FRCNTGB  | Rx Frame Count (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_RXIMSK.FRCNTGB bit, when set, masks the interrupt when rxframecount_gb counter reaches half or full. |

## MMC TX Interrupt Mask Register

The EMAC\_MMC\_TXIMSK register enables (unmasks) MMC transmit interrupts.



# EMAC\_MMC\_TXIMSK: MMC TX Interrupt Mask Register - R/W

Reset = 0x0000 0000

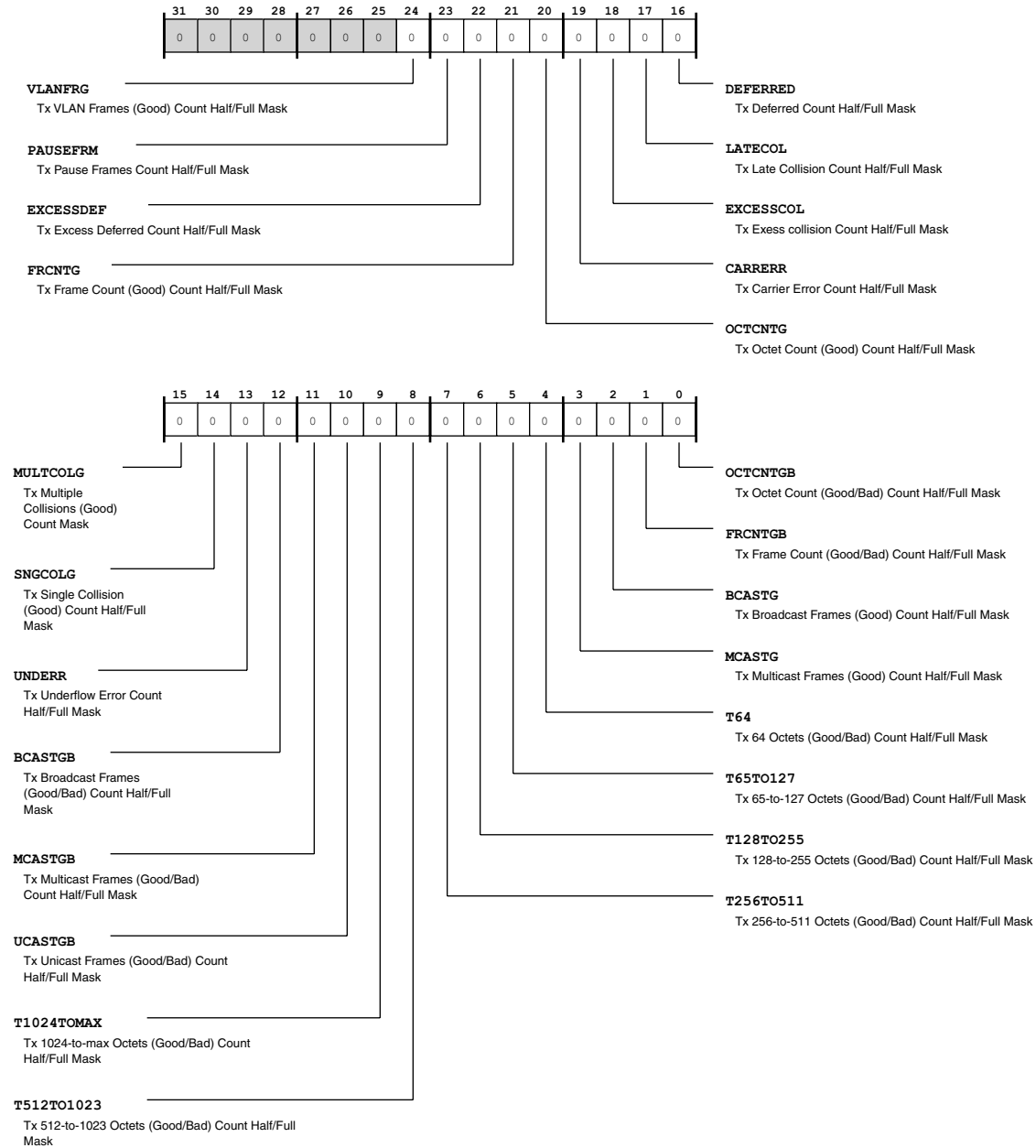


Figure 23-33: EMAC\_MMC\_TXIMSK Register Diagram

**Table 23-57:** EMAC\_MMC\_TXIMSK Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                         |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24      | VLANFRG   | Tx VLAN Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.VLANFRG bit, when set, masks the interrupt when txvlanframes_g counter reaches full or half. |
| 23      | PAUSEFRM  | Tx Pause Frames Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.PAUSEFRM bit, when set, masks the interrupt when txpauseframes counter reaches full or half.       |
| 22      | EXCESSDEF | Tx Excess Deferred Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.EXCESSDEF bit, when set, masks the interrupt when txexcessdef counter reaches full or half.     |
| 21      | FRCNTG    | Tx Frame Count (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.FRCNTG bit, when set, masks the interrupt when txframecount_g counter reaches full or half.  |
| 20      | OCTCNTG   | Tx Octet Count (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.OCTCNTG bit, when set, masks the interrupt when txoctetcount_g counter reaches full or half. |
| 19      | CARRERR   | Tx Carrier Error Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.CARRERR bit, when set, masks the interrupt when txcarriererr counter reaches full or half.        |
| 18      | EXCESSCOL | Tx Excess collision Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.EXCESSCOL bit, when set, masks the interrupt when txexesscol counter reaches full or half.     |
| 17      | LATECOL   | Tx Late Collision Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.LATECOL bit, when set, masks the interrupt when txlatecol counter reaches full or half.          |
| 16      | DEFERRED  | Tx Deferred Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.DEFERRED bit, when set, masks the interrupt when txdeferred counter reaches full or half.              |
| 15      | MULTCOLG  | Tx Multiple Collisions (Good) Count Mask.<br>The EMAC_MMC_TXIMSK.MULTCOLG bit, when set, masks the interrupt when txmulticol_g counter reaches full or half.    |

**Table 23-57:** EMAC\_MMC\_TXIMSK Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                             |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | SNGCOLG    | Tx Single Collision (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.SNGCOLG bit, when set, masks the interrupt when txsinglecol_g counter reaches full or half.                 |
| 13      | UNDERR     | Tx Underflow Error Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.UNDERR bit, when set, masks the interrupt when txunderflowerror counter reaches full or half.                       |
| 12      | BCASTGB    | Tx Broadcast Frames (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.BCASTGB bit, when set, masks the interrupt when txbroadcastframes_gb counter reaches full or half.      |
| 11      | MCASTGB    | Tx Multicast Frames (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.MCASTGB bit, when set, masks the interrupt when txmulticastframes_gb counter reaches full or half.      |
| 10      | UCASTGB    | Tx Unicast Frames (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.UCASTGB bit, when set, masks the interrupt when txunicastframes_gb counter reaches full or half.          |
| 9       | T1024TOMAX | Tx 1024-to-max Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T1024TOMAX bit, when set, masks the interrupt when tx1024tomaxoctets_gb counter reaches full or half. |
| 8       | T512TO1023 | Tx 512-to-1023 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T512TO1023 bit, when set, masks the interrupt when tx512to1023octets_gb counter reaches full or half. |
| 7       | T256TO511  | Tx 256-to-511 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T256TO511 bit, when set, masks the interrupt when tx256to511octets_gb counter reaches full or half.    |
| 6       | T128TO255  | Tx 128-to-255 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T128TO255 bit, when set, masks the interrupt when tx128to255octets_gb counter reaches full or half.    |
| 5       | T65TO127   | Tx 65-to-127 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T65TO127 bit, when set, masks the interrupt when tx65to127octets_gb counter reaches full or half.       |
| 4       | T64        | Tx 64 Octets (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.T64 bit, when set, masks the interrupt when tx64octets_gb counter reaches full or half.                        |

Table 23-57: EMAC\_MMC\_TXIMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | MCASTG   | Tx Multicast Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.MCASTG bit, when set, masks the interrupt when txmulticastframes_g counter reaches full or half. |
| 2       | BCASTG   | Tx Broadcast Frames (Good) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.BCASTG bit, when set, masks the interrupt when txbroadcastframes_g counter reaches full or half. |
| 1       | FRCNTGB  | Tx Frame Count (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.FRCNTGB bit, when set, masks the interrupt when txframecount_gb counter reaches full or half.     |
| 0       | OCTCNTGB | Tx Octet Count (Good/Bad) Count Half/Full Mask.<br>The EMAC_MMC_TXIMSK.OCTCNTGB bit, when set, masks the interrupt when txoctetcount_gb counter reaches full or half.    |

Tx OCT Count (Good/Bad) Register

The EMAC\_TXOCTCNT\_GB register contains the count of the number of bytes transmitted, exclusive of the preamble and retried bytes, in good and bad frames.

EMAC\_TXOCTCNT\_GB: Tx OCT Count (Good/Bad) Register - R/NW

Reset = 0x0000 0000

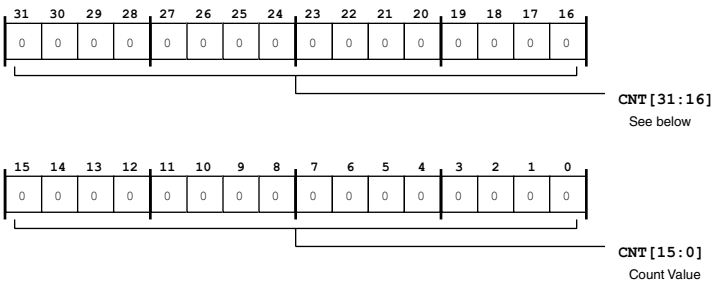


Figure 23-34: EMAC\_TXOCTCNT\_GB Register Diagram

Table 23-58: EMAC\_TXOCTCNT\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Frame Count (Good/Bad) Register

The EMAC\_TXFRMCNT\_GB register contains the count of the number of good and bad frames transmitted, exclusive of retried frames.

EMAC\_TXFRMCNT\_GB: Tx Frame Count (Good/Bad) Register - R/NW

Reset = 0x0000 0000

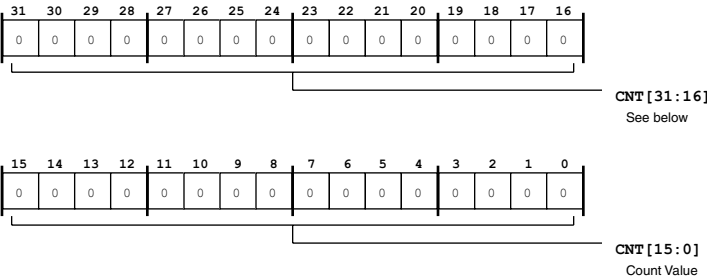


Figure 23-35: EMAC\_TXFRMCNT\_GB Register Diagram

Table 23-59: EMAC\_TXFRMCNT\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Broadcast Frames (Good) Register

The EMAC\_TXBCASTFRM\_G register contains the count of the number of good broadcast frames transmitted.

EMAC\_TXBCASTFRM\_G: Tx Broadcast Frames (Good) Register - R/NW

Reset = 0x0000 0000

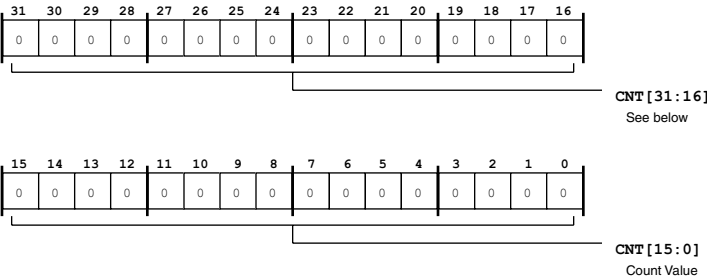


Figure 23-36: EMAC\_TXBCASTFRM\_G Register Diagram

Table 23-60: EMAC\_TXBCASTFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Multicast Frames (Good) Register

The EMAC\_TXMCASTFRM\_G register contains the count of the number of good multicast frames transmitted.

EMAC\_TXMCASTFRM\_G: Tx Multicast Frames (Good) Register - R/NW

Reset = 0x0000 0000

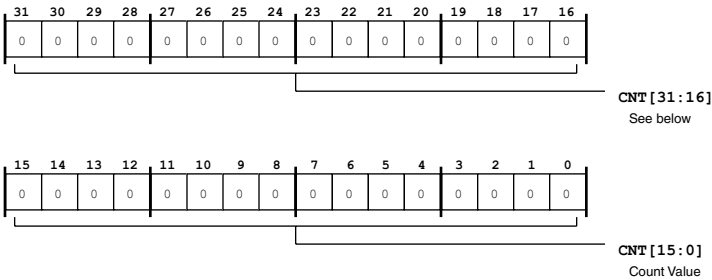


Figure 23-37: EMAC\_TXMCASTFRM\_G Register Diagram

Table 23-61: EMAC\_TXMCASTFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx 64-Byte Frames (Good/Bad) Register

The EMAC\_TX64\_GB register contains the count of the number of good and bad frames transmitted with length 64 bytes, exclusive of preamble and retried frames.

EMAC\_TX64\_GB: Tx 64-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

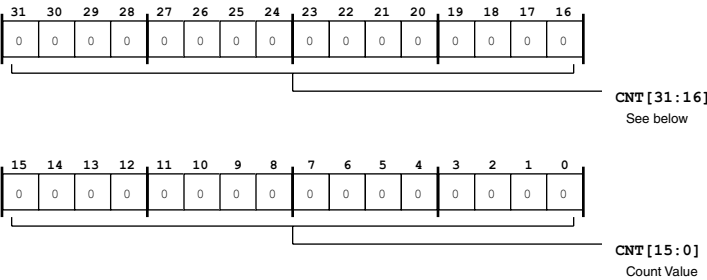


Figure 23-38: EMAC\_TX64\_GB Register Diagram

Table 23-62: EMAC\_TX64\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx 65- to 127-Byte Frames (Good/Bad) Register

The EMAC\_TX65TO127\_GB register contains the count of the number of good and bad frames transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried frames.

EMAC\_TX65TO127\_GB: Tx 65- to 127-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

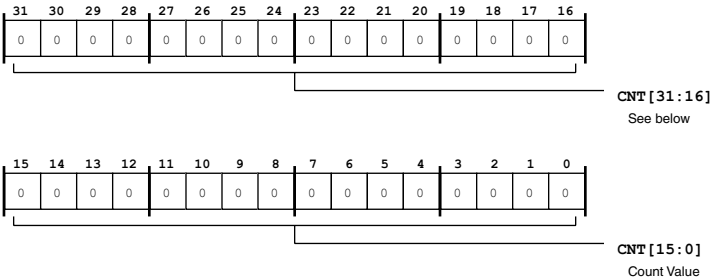


Figure 23-39: EMAC\_TX65TO127\_GB Register Diagram

Table 23-63: EMAC\_TX65TO127\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx 128- to 255-Byte Frames (Good/Bad) Register

The EMAC\_TX128TO255\_GB register contains the count of the number of good and bad frames transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried frames.

EMAC\_TX128TO255\_GB: Tx 128- to 255-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

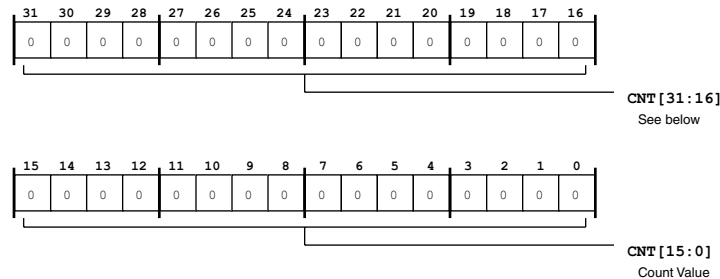


Figure 23-40: EMAC\_TX128TO255\_GB Register Diagram

Table 23-64: EMAC\_TX128TO255\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx 256- to 511-Byte Frames (Good/Bad) Register

The EMAC\_TX256TO511\_GB register contains the count of the number of good and bad frames transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried frames.

EMAC\_TX256TO511\_GB: Tx 256- to 511-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

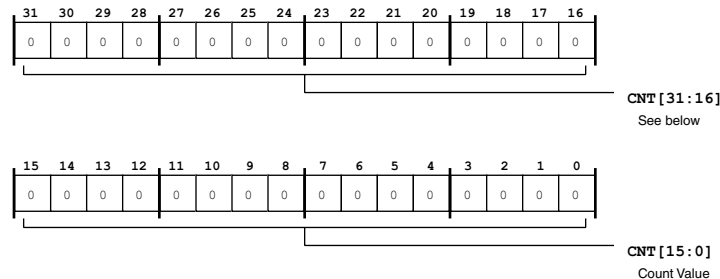


Figure 23-41: EMAC\_TX256TO511\_GB Register Diagram



**Table 23-65:** EMAC\_TX256TO511\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx 512- to 1023-Byte Frames (Good/Bad) Register

The EMAC\_TX512TO1023\_GB register contains the count of the number of good and bad frames transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried frames.

**EMAC\_TX512TO1023\_GB: Tx 512- to 1023-Byte Frames (Good/Bad) Register - R/NW**

Reset = 0x0000 0000

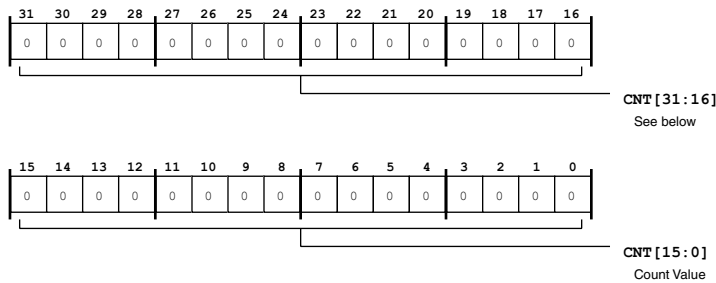


**Figure 23-42:** EMAC\_TX512TO1023\_GB Register Diagram

**Table 23-66:** EMAC\_TX512TO1023\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx 1024- to Max-Byte Frames (Good/Bad) Register

The EMAC\_TX1024TOMAX\_GB register contains the count of the number of good and bad frames transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried frames.

EMAC\_TX1024TOMAX\_GB: Tx 1024- to Max-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

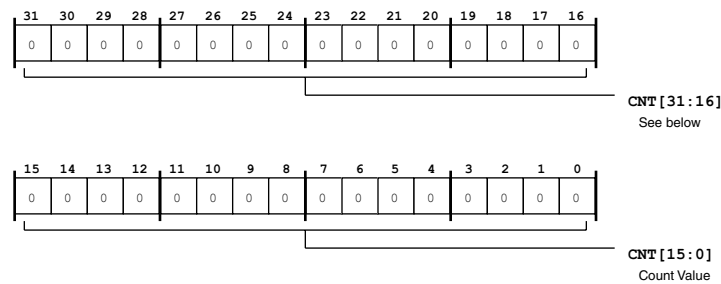


Figure 23-43: EMAC\_TX1024TOMAX\_GB Register Diagram

Table 23-67: EMAC\_TX1024TOMAX\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Unicast Frames (Good/Bad) Register

The EMAC\_TXUCASTFRM\_GB register contains the count of the number of good and bad unicast frames transmitted.

EMAC\_TXUCASTFRM\_GB: Tx Unicast Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

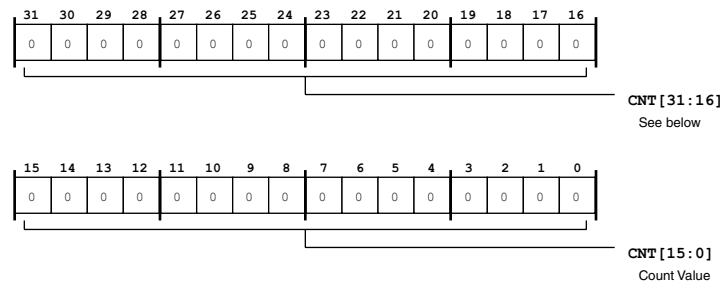


Figure 23-44: EMAC\_TXUCASTFRM\_GB Register Diagram

Table 23-68: EMAC\_TXUCASTFRM\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Multicast Frames (Good/Bad) Register

The EMAC\_TXMCASTFRM\_GB register contains the count of the number of good and bad multicast frames transmitted.

EMAC\_TXMCASTFRM\_GB: Tx Multicast Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

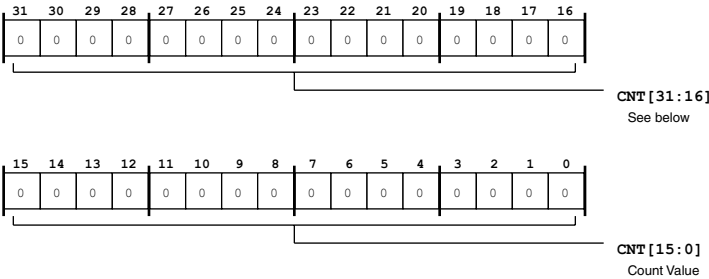


Figure 23-45: EMAC\_TXMCASTFRM\_GB Register Diagram

Table 23-69: EMAC\_TXMCASTFRM\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Broadcast Frames (Good/Bad) Register

The EMAC\_TXBCASTFRM\_GB register contains the count of the number of good and bad broadcast frames transmitted.

EMAC\_TXBCASTFRM\_GB: Tx Broadcast Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

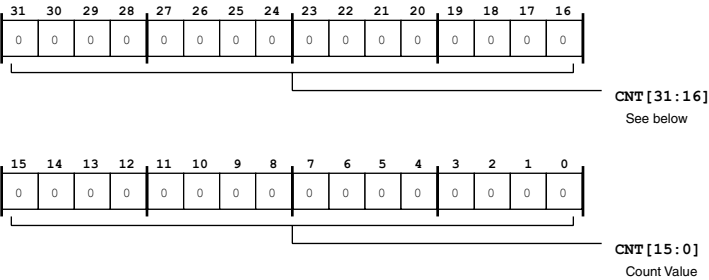


Figure 23-46: EMAC\_TXBCASTFRM\_GB Register Diagram

Table 23-70: EMAC\_TXBCASTFRM\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Underflow Error Register

The EMAC\_TXUNDR\_ERR register contains a count of the number of frames aborted due to frame underflow error.

EMAC\_TXUNDR\_ERR: Tx Underflow Error Register - R/NW

Reset = 0x0000 0000

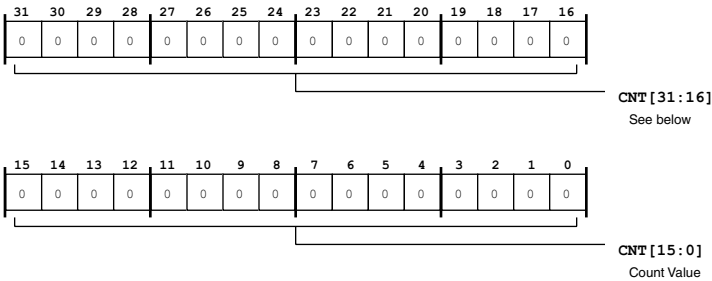


Figure 23-47: EMAC\_TXUNDR\_ERR Register Diagram

Table 23-71: EMAC\_TXUNDR\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Single Collision (Good) Register

The EMAC\_TXSNGCOL\_G register contains a count of the number of successfully transmitted frames after a single collision in Half-duplex mode.

EMAC\_TXSNGCOL\_G: Tx Single Collision (Good) Register - R/NW

Reset = 0x0000 0000

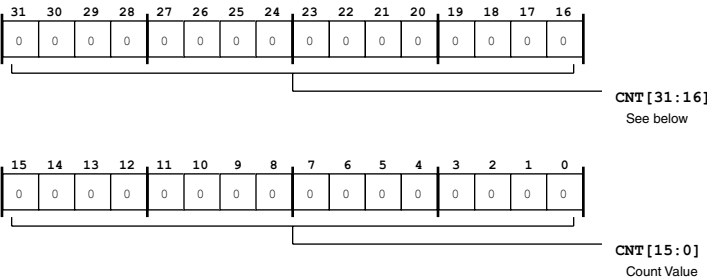


Figure 23-48: EMAC\_TXSNGCOL\_G Register Diagram

Table 23-72: EMAC\_TXSNGCOL\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Multiple Collision (Good) Register

The EMAC\_TXMULTCOL\_G register contains a count of the number of successfully transmitted frames after more than a single collision in Half-duplex mode.

EMAC\_TXMULTCOL\_G: Tx Multiple Collision (Good) Register - R/NW

Reset = 0x0000 0000

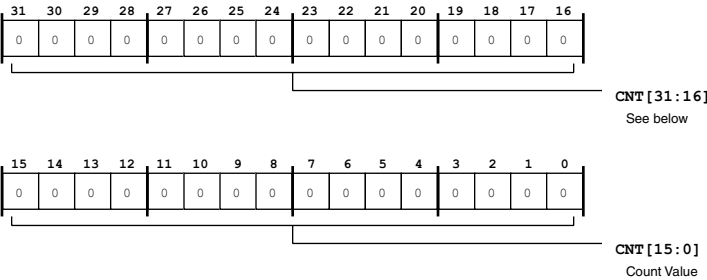


Figure 23-49: EMAC\_TXMULTCOL\_G Register Diagram

Table 23-73: EMAC\_TXMULTCOL\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Deferred Register

The EMAC\_TXDEFERRED register contains a count of the number of successfully transmitted frames after a deferral in Half-duplex mode.

**EMAC\_TXDEFERRED: Tx Deferred Register - R/NW**

Reset = 0x0000 0000

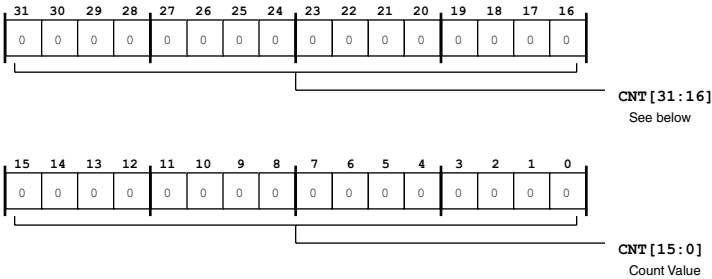


Figure 23-50: EMAC\_TXDEFERRED Register Diagram

Table 23-74: EMAC\_TXDEFERRED Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Late Collision Register

The EMAC\_TXLATECOL register contains a count of the number of frames aborted due to late collision error.

**EMAC\_TXLATECOL: Tx Late Collision Register - R/NW**

Reset = 0x0000 0000

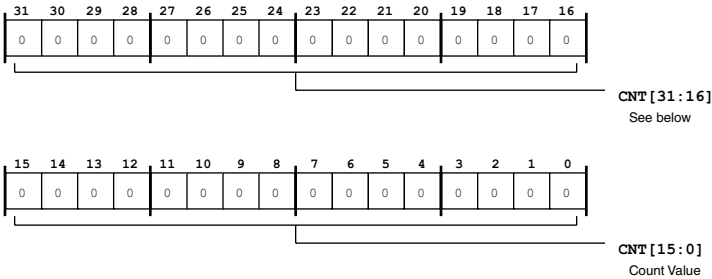


Figure 23-51: EMAC\_TXLATECOL Register Diagram

**Table 23-75:** EMAC\_TXLATECOL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

**Tx Excess Collision Register**

The EMAC\_TXEXCESSCOL register contains a count of the number of frames aborted due to excessive (16) collision errors.

**EMAC\_TXEXCESSCOL: Tx Excess Collision Register - R/NW**

Reset = 0x0000 0000

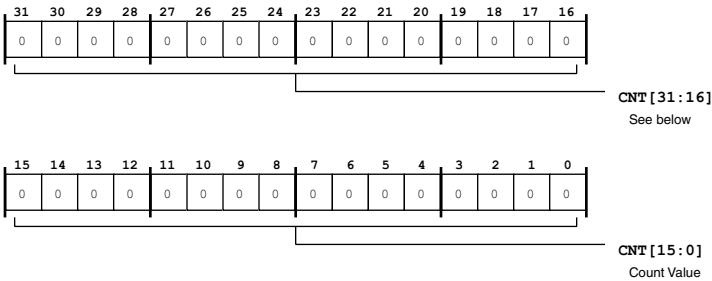


**Figure 23-52:** EMAC\_TXEXCESSCOL Register Diagram

**Table 23-76:** EMAC\_TXEXCESSCOL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

**Tx Carrier Error Register**

The EMAC\_TXCARR\_ERR register contains a count of the number of frames aborted due to carrier sense error (no carrier or loss of carrier).

EMAC\_TXCARR\_ERR: Tx Carrier Error Register - R/NW

Reset = 0x0000 0000

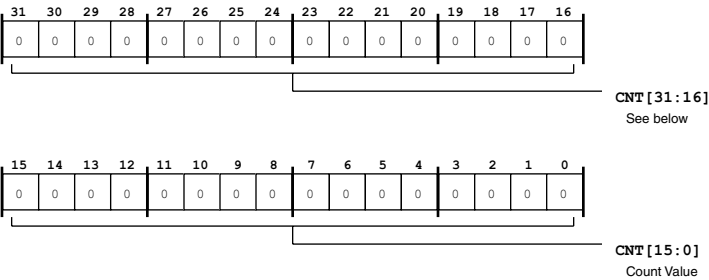


Figure 23-53: EMAC\_TXCARR\_ERR Register Diagram

Table 23-77: EMAC\_TXCARR\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Octet Count (Good) Register

The EMAC\_TXOCTCNT\_G register contains a count of the number of bytes transmitted, exclusive of preamble, in good frames only.

EMAC\_TXOCTCNT\_G: Tx Octet Count (Good) Register - R/NW

Reset = 0x0000 0000

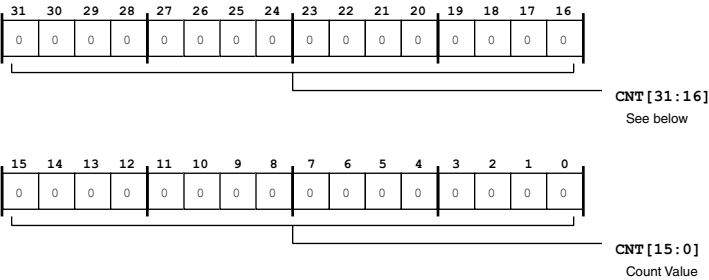


Figure 23-54: EMAC\_TXOCTCNT\_G Register Diagram

Table 23-78: EMAC\_TXOCTCNT\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |



## Tx Frame Count (Good) Register

The EMAC\_TXFRMCNT\_G register contains a count of the number of good frames transmitted.

EMAC\_TXFRMCNT\_G: Tx Frame Count (Good) Register - R/NW

Reset = 0x0000 0000

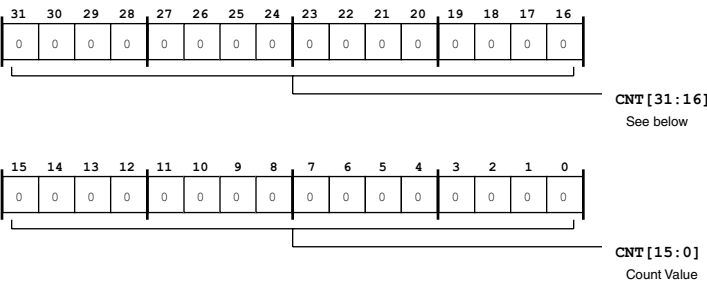


Figure 23-55: EMAC\_TXFRMCNT\_G Register Diagram

Table 23-79: EMAC\_TXFRMCNT\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Tx Excess Deferral Register

The EMAC\_TXEXCESSDEF register contains a count of the number of frames aborted due to excessive deferral error (deferred for more than two max-sized frame times).

EMAC\_TXEXCESSDEF: Tx Excess Deferral Register - R/NW

Reset = 0x0000 0000

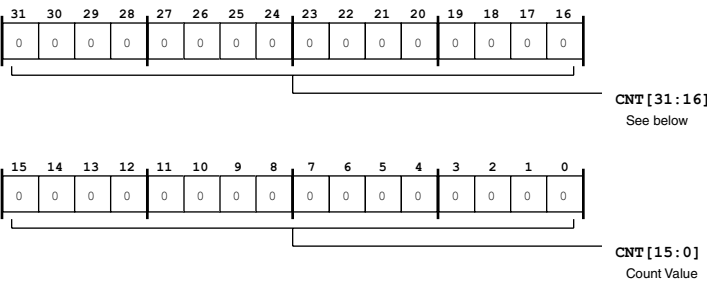


Figure 23-56: EMAC\_TXEXCESSDEF Register Diagram

**Table 23-80:** EMAC\_TXEXCESSDEF Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx Pause Frame Register

The EMAC\_TXPAUSEFRM register contains a count of the number of good PAUSE frames transmitted.

**EMAC\_TXPAUSEFRM: Tx Pause Frame Register - R/NW**

Reset = 0x0000 0000

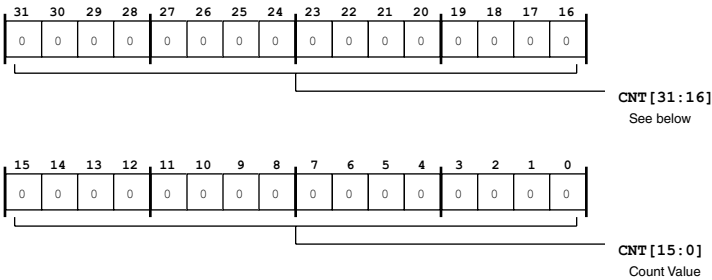


**Figure 23-57:** EMAC\_TXPAUSEFRM Register Diagram

**Table 23-81:** EMAC\_TXPAUSEFRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Tx VLAN Frames (Good) Register

The EMAC\_TXVLANFRM\_G register contains a count of the number of good VLAN frames transmitted, exclusive of retried frames.

EMAC\_TXVLANFRM\_G: Tx VLAN Frames (Good) Register - R/NW

Reset = 0x0000 0000

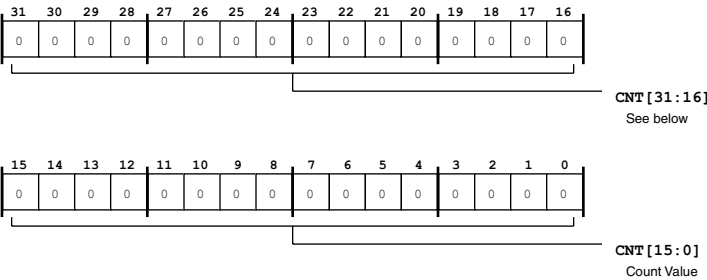


Figure 23-58: EMAC\_TXVLANFRM\_G Register Diagram

Table 23-82: EMAC\_TXVLANFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Frame Count (Good/Bad) Register

The EMAC\_RXFRMCNT\_GB register contains a count of the number of good and bad frames received.

EMAC\_RXFRMCNT\_GB: Rx Frame Count (Good/Bad) Register - R/NW

Reset = 0x0000 0000

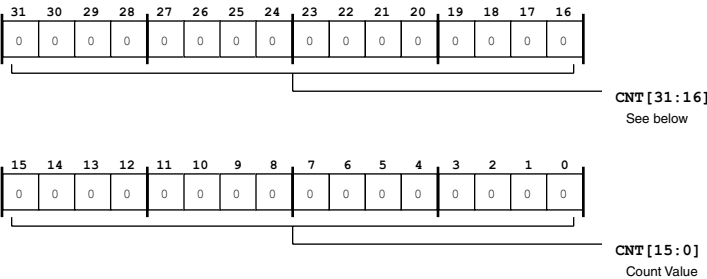


Figure 23-59: EMAC\_RXFRMCNT\_GB Register Diagram

Table 23-83: EMAC\_RXFRMCNT\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx Octet Count (Good/Bad) Register

The EMAC\_RXOCTCNT\_GB register contains a count of the number of bytes received, exclusive of preamble, in good and bad frames.

EMAC\_RXOCTCNT\_GB: Rx Octet Count (Good/Bad) Register - R/NW

Reset = 0x0000 0000

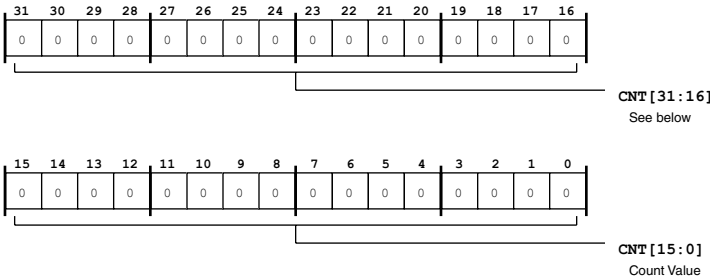


Figure 23-60: EMAC\_RXOCTCNT\_GB Register Diagram

Table 23-84: EMAC\_RXOCTCNT\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx Octet Count (Good) Register

The EMAC\_RXOCTCNT\_G register contains a count of the number of bytes received, exclusive of preamble, only in good frames.

EMAC\_RXOCTCNT\_G: Rx Octet Count (Good) Register - R/NW

Reset = 0x0000 0000

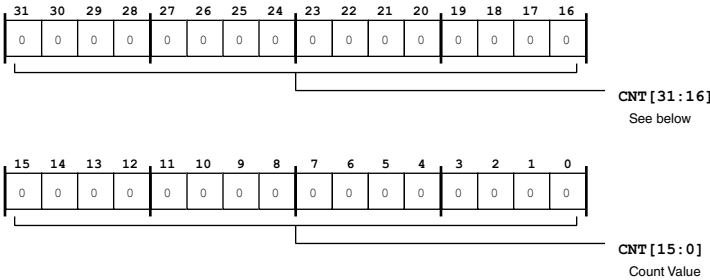


Figure 23-61: EMAC\_RXOCTCNT\_G Register Diagram

**Table 23-85:** EMAC\_RXOCTCNT\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Broadcast Frames (Good) Register

The EMAC\_RXBCASTFRM\_G register contains a count of the number of good broadcast frames received.

**EMAC\_RXBCASTFRM\_G: Rx Broadcast Frames (Good) Register - R/NW**

Reset = 0x0000 0000

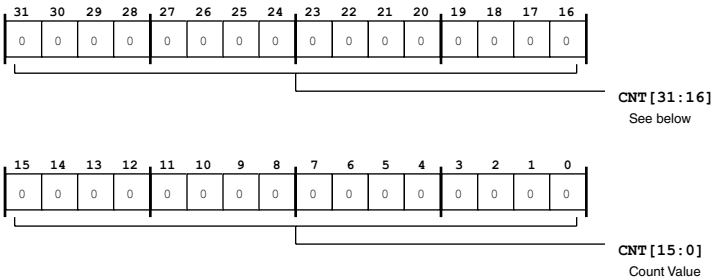


**Figure 23-62:** EMAC\_RXBCASTFRM\_G Register Diagram

**Table 23-86:** EMAC\_RXBCASTFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Multicast Frames (Good) Register

The EMAC\_RXMCASTFRM\_G register contains a count of the number of good multicast frames received.

EMAC\_RXMCASTFRM\_G: Rx Multicast Frames (Good) Register - R/NW

Reset = 0x0000 0000

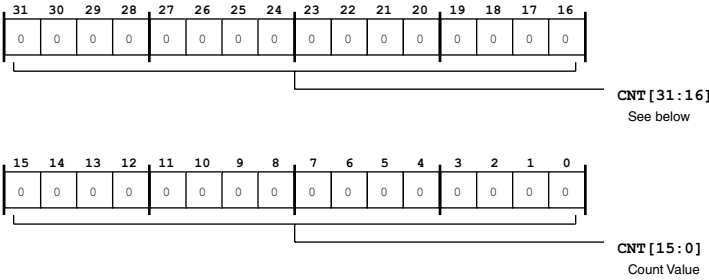


Figure 23-63: EMAC\_RXMCASTFRM\_G Register Diagram

Table 23-87: EMAC\_RXMCASTFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx CRC Error Register

The EMAC\_RXCRC\_ERR register contains a count of the number of frames received with CRC error.

EMAC\_RXCRC\_ERR: Rx CRC Error Register - R/NW

Reset = 0x0000 0000

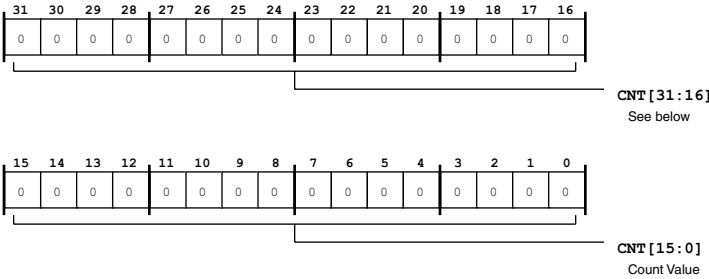


Figure 23-64: EMAC\_RXCRC\_ERR Register Diagram

Table 23-88: EMAC\_RXCRC\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx alignment Error Register

The EMAC\_RXALIGN\_ERR register contains a count of the number of frames received with alignment (dribble) error. Valid only in 10/100 mode.

EMAC\_RXALIGN\_ERR: Rx alignment Error Register - R/NW

Reset = 0x0000 0000

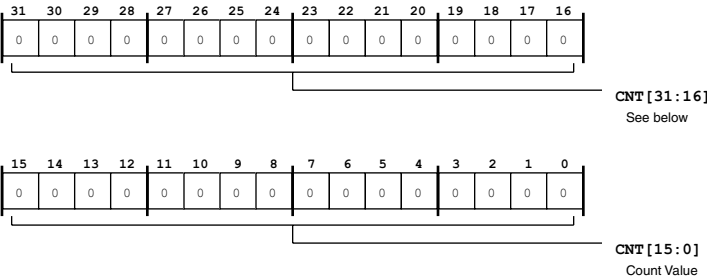


Figure 23-65: EMAC\_RXALIGN\_ERR Register Diagram

Table 23-89: EMAC\_RXALIGN\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx Runt Error Register

The EMAC\_RXRUNT\_ERR register contains a count of the number of frames received with runt error.

EMAC\_RXRUNT\_ERR: Rx Runt Error Register - R/NW

Reset = 0x0000 0000

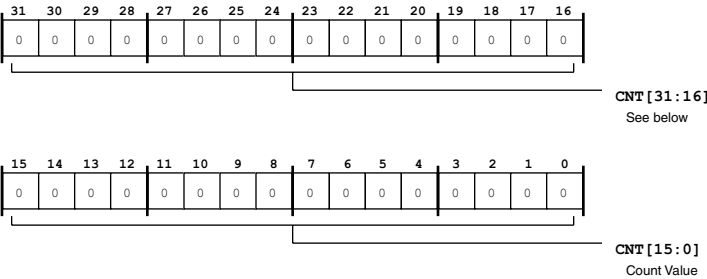


Figure 23-66: EMAC\_RXRUNT\_ERR Register Diagram

**Table 23-90:** EMAC\_RXRUNT\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Jab Error Register

The EMAC\_RXJAB\_ERR register contains a count of the number of giant frames received with length greater than 1,518 bytes and with CRC error.

**EMAC\_RXJAB\_ERR: Rx Jab Error Register - R/NW**

Reset = 0x0000 0000

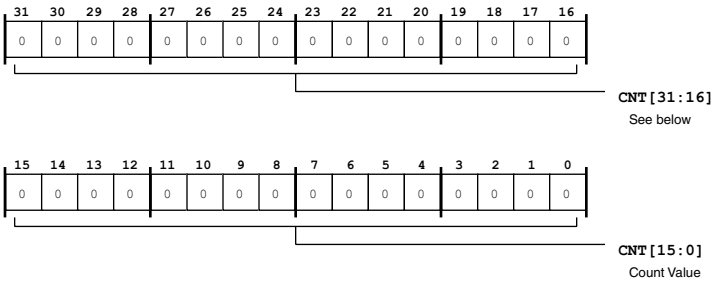


**Figure 23-67:** EMAC\_RXJAB\_ERR Register Diagram

**Table 23-91:** EMAC\_RXJAB\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Undersize (Good) Register

The EMAC\_RXUSIZE\_G register contains a count of the number of frames received with length less than 64 bytes, without any errors.



EMAC\_RXUSIZE\_G: Rx Undersize (Good) Register - R/NW

Reset = 0x0000 0000

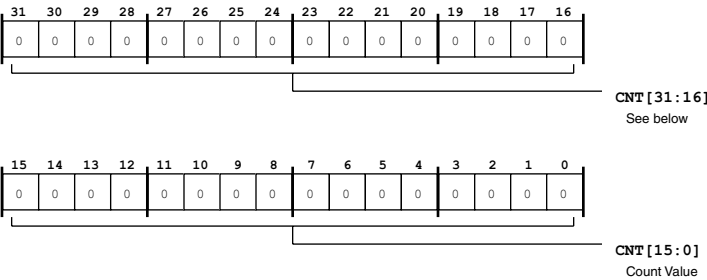


Figure 23-68: EMAC\_RXUSIZE\_G Register Diagram

Table 23-92: EMAC\_RXUSIZE\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Oversize (Good) Register

The EMAC\_RXOSIZE\_G register contains a count of the number of frames received with length greater than the maxsize, without errors.

EMAC\_RXOSIZE\_G: Rx Oversize (Good) Register - R/NW

Reset = 0x0000 0000

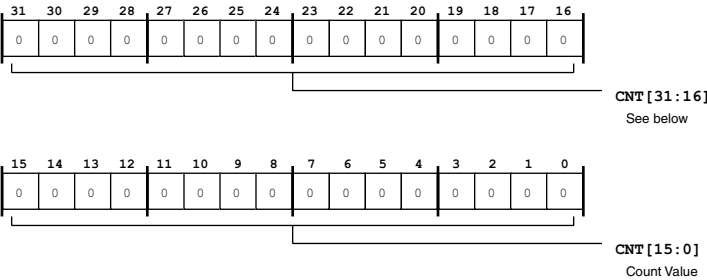


Figure 23-69: EMAC\_RXOSIZE\_G Register Diagram

Table 23-93: EMAC\_RXOSIZE\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx 64-Byte Frames (Good/Bad) Register

The EMAC\_RX64\_GB register contains a count of the number of good and bad frames received with length 64 bytes, exclusive of preamble.

EMAC\_RX64\_GB: Rx 64-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

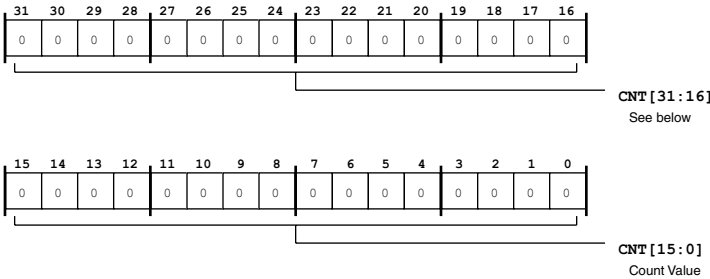


Figure 23-70: EMAC\_RX64\_GB Register Diagram

Table 23-94: EMAC\_RX64\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx 65- to 127-Byte Frames (Good/Bad) Register

The EMAC\_RX65TO127\_GB register contains a count of the number of good and bad frames received with length between 65 and 127 (inclusive) bytes, exclusive of preamble.

EMAC\_RX65TO127\_GB: Rx 65- to 127-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

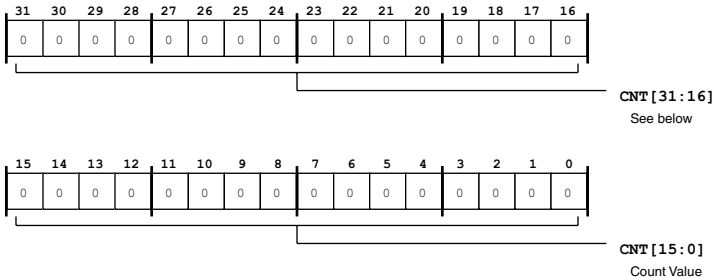


Figure 23-71: EMAC\_RX65TO127\_GB Register Diagram

**Table 23-95:** EMAC\_RX65TO127\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx 128- to 255-Byte Frames (Good/Bad) Register

The EMAC\_RX128TO255\_GB register contains a count of the number of good and bad frames received with length between 128 and 255 (inclusive) bytes, exclusive of preamble.

**EMAC\_RX128TO255\_GB: Rx 128- to 255-Byte Frames (Good/Bad) Register - R/NW**

Reset = 0x0000 0000

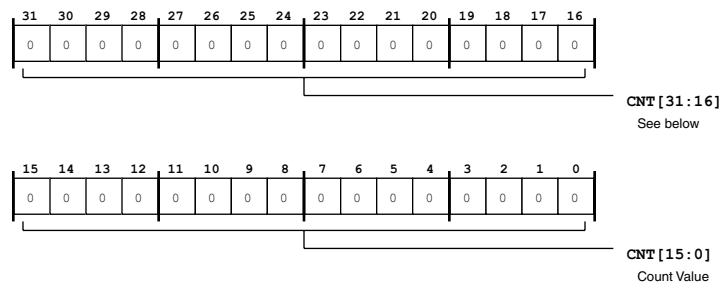


**Figure 23-72:** EMAC\_RX128TO255\_GB Register Diagram

**Table 23-96:** EMAC\_RX128TO255\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx 256- to 511-Byte Frames (Good/Bad) Register

The EMAC\_RX256TO511\_GB register contains a count of the number of good and bad frames received with length between 256 and 511 (inclusive) bytes, exclusive of preamble.

EMAC\_RX256TO511\_GB: Rx 256- to 511-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

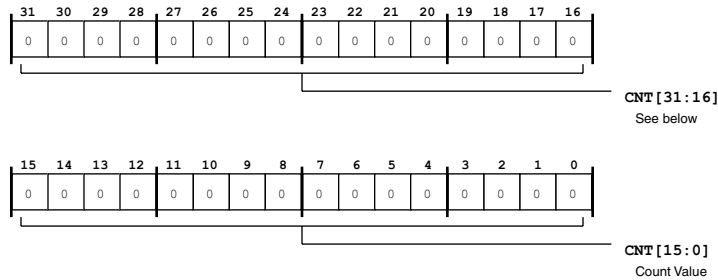


Figure 23-73: EMAC\_RX256TO511\_GB Register Diagram

Table 23-97: EMAC\_RX256TO511\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx 512- to 1023-Byte Frames (Good/Bad) Register

The EMAC\_RX512TO1023\_GB register contains a count of the number of good and bad frames received with length between 512 and 1023 (inclusive) bytes, exclusive of preamble.

EMAC\_RX512TO1023\_GB: Rx 512- to 1023-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

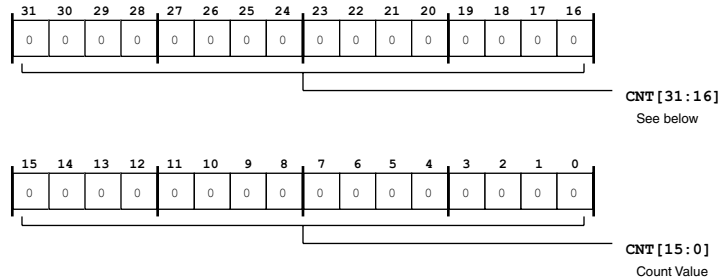


Figure 23-74: EMAC\_RX512TO1023\_GB Register Diagram

Table 23-98: EMAC\_RX512TO1023\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx 1024- to Max-Byte Frames (Good/Bad) Register

The EMAC\_RX1024TOMAX\_GB register contains a count of the number of good and bad frames received with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble.

EMAC\_RX1024TOMAX\_GB: Rx 1024- to Max-Byte Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

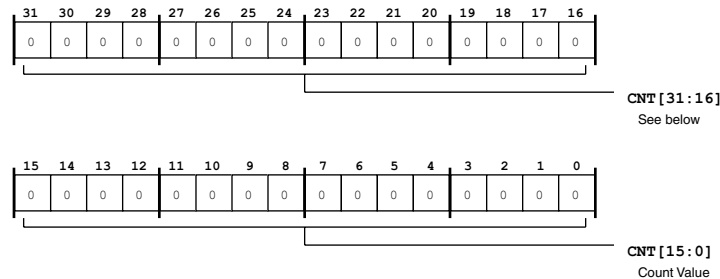


Figure 23-75: EMAC\_RX1024TOMAX\_GB Register Diagram

Table 23-99: EMAC\_RX1024TOMAX\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx Unicast Frames (Good) Register

The EMAC\_RXUNICASTFRM\_G register contains a count of the number of good unicast frames received.

EMAC\_RXUNICASTFRM\_G: Rx Unicast Frames (Good) Register - R/NW

Reset = 0x0000 0000

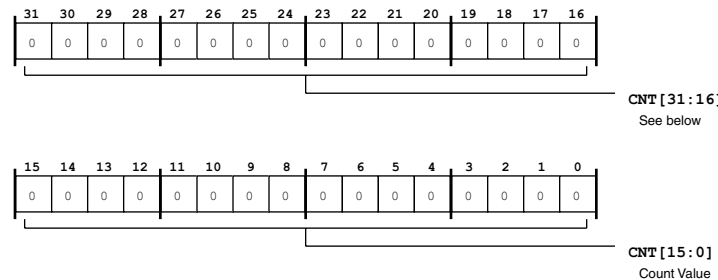


Figure 23-76: EMAC\_RXUNICASTFRM\_G Register Diagram

Table 23-100: EMAC\_RXUCASTFRM\_G Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Length Error Register

The EMAC\_RXLEN\_ERR register contains a count of the number of frames received with length error (Length type field frame size), for all frames with valid length field.

EMAC\_RXLEN\_ERR: Rx Length Error Register - R/NW

Reset = 0x0000 0000

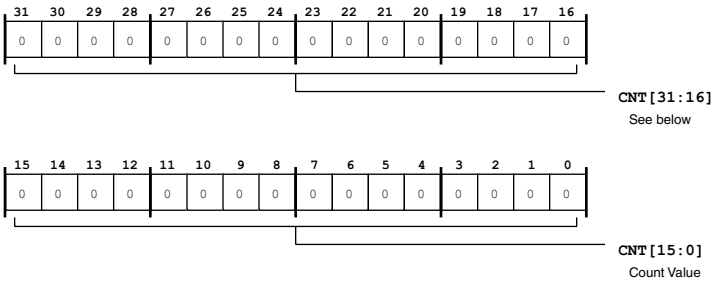


Figure 23-77: EMAC\_RXLEN\_ERR Register Diagram

Table 23-101: EMAC\_RXLEN\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Out Of Range Type Register

The EMAC\_RXOORTYPE register contains a count of the number of frames received with length field not equal to the valid frame size (greater than 1,500 but less than 1,536).

EMAC\_RXOORTYPE: Rx Out Of Range Type Register - R/NW

Reset = 0x0000 0000

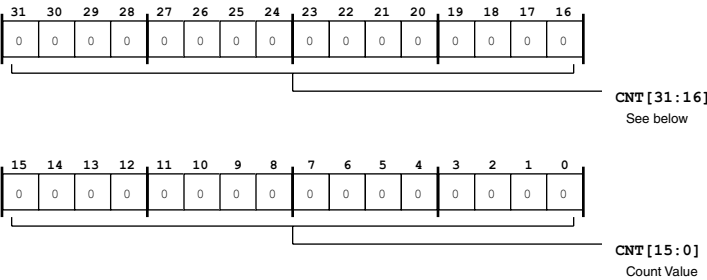


Figure 23-78: EMAC\_RXOORTYPE Register Diagram

Table 23-102: EMAC\_RXOORTYPE Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Pause Frames Register

The EMAC\_RXPAUSEFRM register contains a count of the number of good and valid PAUSE frames received.

EMAC\_RXPAUSEFRM: Rx Pause Frames Register - R/NW

Reset = 0x0000 0000

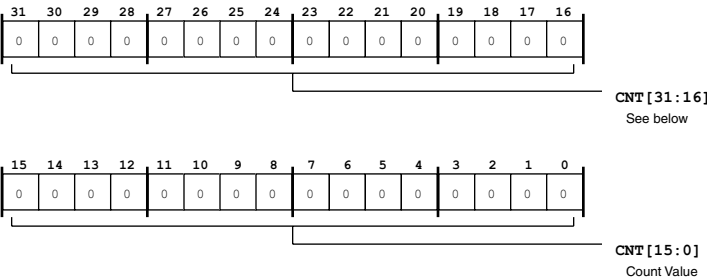


Figure 23-79: EMAC\_RXPAUSEFRM Register Diagram

Table 23-103: EMAC\_RXPAUSEFRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx FIFO Overflow Register

The EMAC\_RXFIFO\_OVF register contains a count of the number of missed received frames due to FIFO overflow.

EMAC\_RXFIFO\_OVF: Rx FIFO Overflow Register - R/NW

Reset = 0x0000 0000

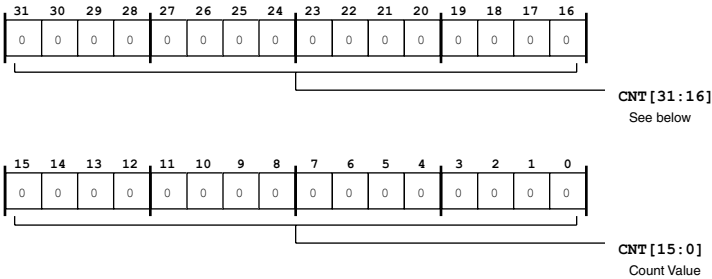


Figure 23-80: EMAC\_RXFIFO\_OVF Register Diagram

Table 23-104: EMAC\_RXFIFO\_OVF Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx VLAN Frames (Good/Bad) Register

The EMAC\_RXVLANFRM\_GB register contains a count of the number of good and bad VLAN frames received.

EMAC\_RXVLANFRM\_GB: Rx VLAN Frames (Good/Bad) Register - R/NW

Reset = 0x0000 0000

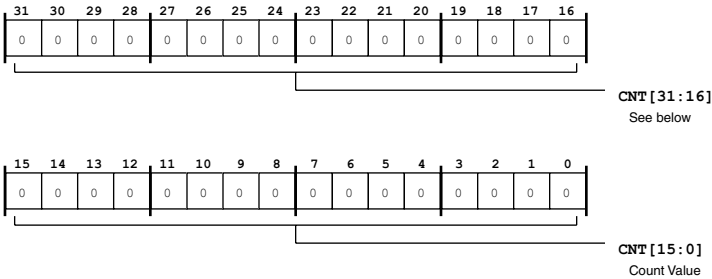


Figure 23-81: EMAC\_RXVLANFRM\_GB Register Diagram



**Table 23-105:** EMAC\_RXVLANFRM\_GB Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx Watch Dog Error Register

The EMAC\_RXWDOG\_ERR register contains a count of the number of frames received with error due to watchdog timeout error (frames with a data load larger than 2,048 bytes).

**EMAC\_RXWDOG\_ERR: Rx Watch Dog Error Register - R/NW**

Reset = 0x0000 0000

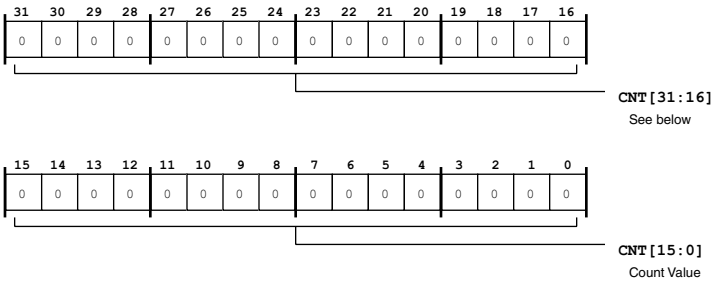


**Figure 23-82:** EMAC\_RXWDOG\_ERR Register Diagram

**Table 23-106:** EMAC\_RXWDOG\_ERR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

MMC IPC Rx Interrupt Mask Register

The EMAC\_IPC\_RXIMSK register enables (unmasks) MMC IPC receive interrupts.

EMAC\_IPC\_RXIMSK: MMC IPC Rx Interrupt Mask Register - R/W

Reset = 0x0000 0000

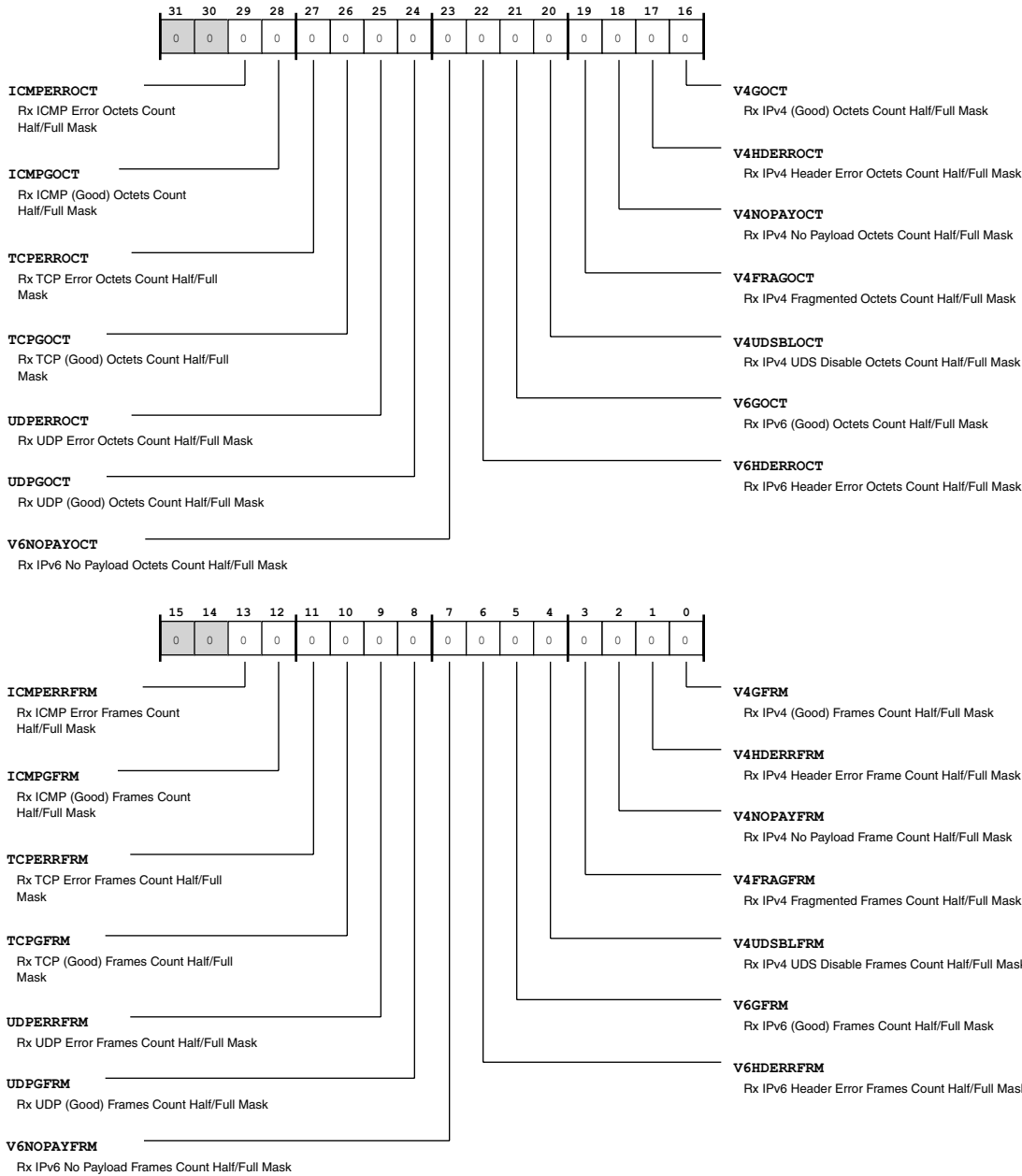


Figure 23-83: EMAC\_IPC\_RXIMSK Register Diagram

**Table 23-107:** EMAC\_IPC\_RXIMSK Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                  |
|---------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29      | ICMPERROCT | Rx ICMP Error Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.ICMPERROCT bit, when set, masks the interrupt when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.           |
| 28      | ICMPGOCT   | Rx ICMP (Good) Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.ICMPGOCT bit, when set, masks the interrupt when the rxicmp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 27      | TCPPERROCT | Rx TCP Error Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.TCPPERROCT bit, when set, masks the interrupt when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 26      | TCPGOCT    | Rx TCP (Good) Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.TCPGOCT bit, when set, masks the interrupt when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 25      | UDPPERROCT | Rx UDP Error Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.UDPPERROCT bit, when set, masks the interrupt when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 24      | UDPGOCT    | Rx UDP (Good) Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.UDPGOCT bit, when set, masks the interrupt when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 23      | V6NOPAYOCT | Rx IPv6 No Payload Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6NOPAYOCT bit, when set, masks the interrupt when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 22      | V6HDERROCT | Rx IPv6 Header Error Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6HDERROCT bit, when set, masks the interrupt when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value. |

**Table 23-107:** EMAC\_IPC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                  |
|---------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 21      | V6GOCT     | Rx IPv6 (Good) Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6GOCT bit, when set, masks the interrupt when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 20      | V4UDSBLOCT | Rx IPv4 UDS Disable Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4UDSBLOCT bit, when set, masks the interrupt when the rxipv4_udsbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.   |
| 19      | V4FRAGOCT  | Rx IPv4 Fragmented Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4FRAGOCT bit, when set, masks the interrupt when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.      |
| 18      | V4NOPAYOCT | Rx IPv4 No Payload Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4NOPAYOCT bit, when set, masks the interrupt when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 17      | V4HDERROCT | Rx IPv4 Header Error Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4HDERROCT bit, when set, masks the interrupt when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value. |
| 16      | V4GOCT     | Rx IPv4 (Good) Octets Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4GOCT bit, when set, masks the interrupt when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 13      | ICMPERRFRM | Rx ICMP Error Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.ICMPERRFRM bit, when set, masks the interrupt when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 12      | ICMPGFRM   | Rx ICMP (Good) Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.ICMPGFRM bit, when set, masks the interrupt when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |

**Table 23-107:** EMAC\_IPC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | TCPERRFRM  | Rx TCP Error Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.TCPERRFRM bit, when set, masks the interrupt when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.              |
| 10      | TCPGFRM    | Rx TCP (Good) Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.TCPGFRM bit, when set, masks the interrupt when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 9       | UDPERFRM   | Rx UDP Error Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.UDPERFRM bit, when set, masks the interrupt when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 8       | UDPGFRM    | Rx UDP (Good) Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.UDPGFRM bit, when set, masks the interrupt when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 7       | V6NOPAYFRM | Rx IPv6 No Payload Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6NOPAYFRM bit, when set, masks the interrupt when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 6       | V6HDERRFRM | Rx IPv6 Header Error Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6HDERRFRM bit, when set, masks the interrupt when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value. |
| 5       | V6GFRM     | Rx IPv6 (Good) Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V6GFRM bit, when set, masks the interrupt when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 4       | V4UDSBLFRM | Rx IPv4 UDS Disable Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4UDSBLFRM bit, when set, masks the interrupt when the rxipv4_udsbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.   |

**Table 23-107:** EMAC\_IPC\_RXIMSK Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                               |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | V4FRAGFRM  | Rx IPv4 Fragmented Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4FRAGFRM bit, when set, masks the interrupt when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.     |
| 2       | V4NOPAYFRM | Rx IPv4 No Payload Frame Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4NOPAYFRM bit, when set, masks the interrupt when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 1       | V4HDERRFRM | Rx IPv4 Header Error Frame Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4HDERRFRM bit, when set, masks the interrupt when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value. |
| 0       | V4GFRM     | Rx IPv4 (Good) Frames Count Half/Full Mask.<br>The EMAC_IPC_RXIMSK.V4GFRM bit, when set, masks the interrupt when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.              |

## MMC IPC Rx Interrupt Register

The EMAC\_IPC\_RXINT register indicates status of MMC IPC receive interrupts.

# EMAC\_IPC\_RXINT: MMC IPC Rx Interrupt Register - R/W

Reset = 0x0000 0000

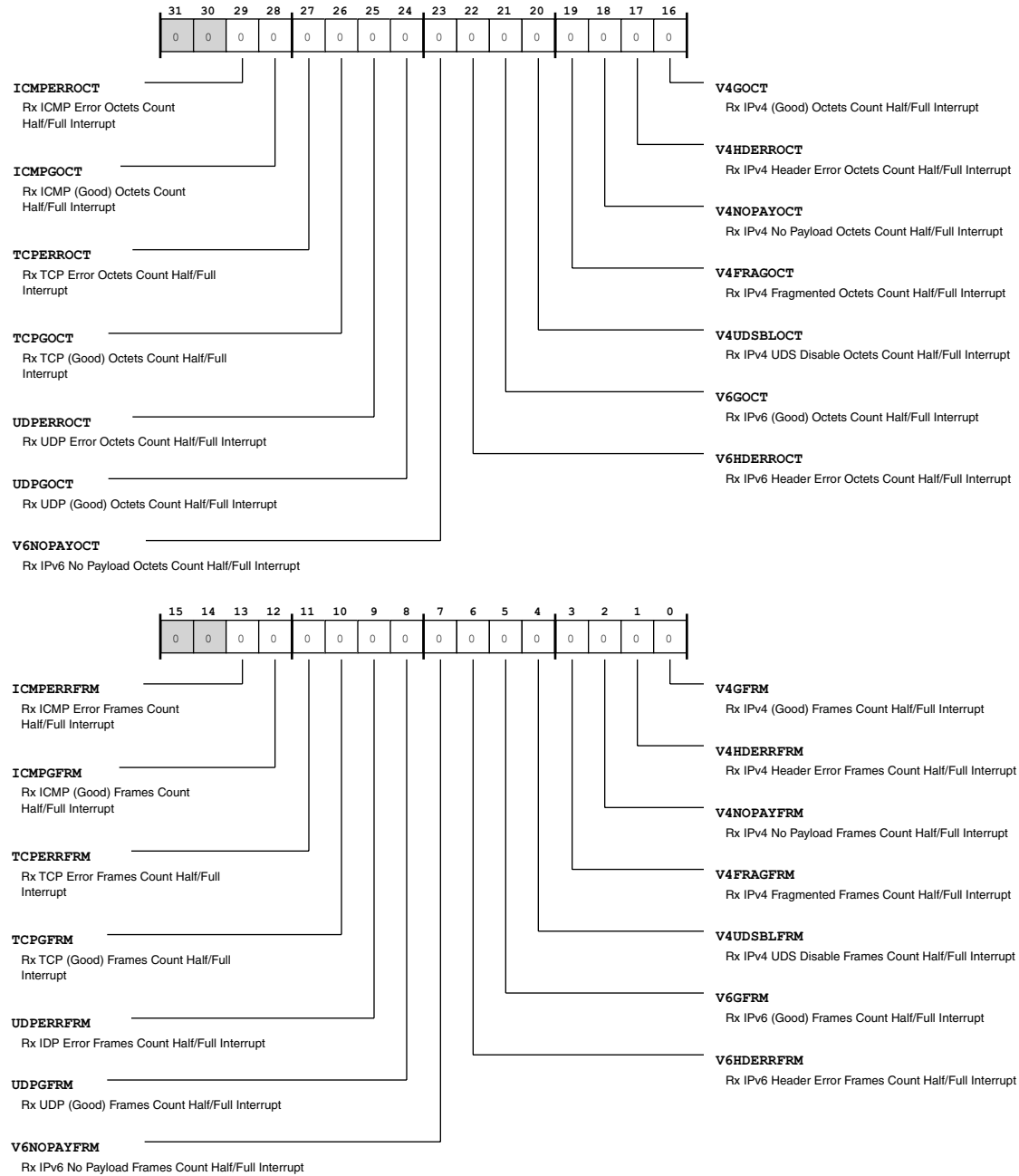


Figure 23-84: EMAC\_IPC\_RXINT Register Diagram

**Table 23-108:** EMAC\_IPC\_RXINT Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                              |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29      | ICMPERROCT | Rx ICMP Error Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.ICMPERROCT bit is set when the rxicmp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.           |
| 28      | ICMPGOCT   | Rx ICMP (Good) Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.ICMPGOCT bit is set when the rxicmp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 27      | TCPPERROCT | Rx TCP Error Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.TCPPERROCT bit is set when the rxtcp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 26      | TCPGOCT    | Rx TCP (Good) Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.TCPGOCT bit is set when the rxtcp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 25      | UDPPERROCT | Rx UDP Error Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.UDPPERROCT bit is set when the rxudp_err_octets counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 24      | UDPGOCT    | Rx UDP (Good) Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.UDPGOCT bit is set when the rxudp_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 23      | V6NOPAYOCT | Rx IPv6 No Payload Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6NOPAYOCT bit is set when the rxipv6_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 22      | V6HDERROCT | Rx IPv6 Header Error Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6HDERROCT bit is set when the rxipv6_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value. |



**Table 23-108:** EMAC\_IPC\_RXINT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                              |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 21      | V6GOCT     | Rx IPv6 (Good) Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6GOCT bit is set when the rxipv6_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 20      | V4UDSBLOCT | Rx IPv4 UDS Disable Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4UDSBLOCT bit is set when the rxipv4_udtbl_octets counter reaches half the maximum value, and also when it reaches the maximum value.   |
| 19      | V4FRAGOCT  | Rx IPv4 Fragmented Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4FRAGOCT bit is set when the rxipv4_frag_octets counter reaches half the maximum value, and also when it reaches the maximum value.      |
| 18      | V4NOPAYOCT | Rx IPv4 No Payload Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4NOPAYOCT bit set when the rxipv4_nopay_octets counter reaches half the maximum value, and also when it reaches the maximum value.       |
| 17      | V4HDERROCT | Rx IPv4 Header Error Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4HDERROCT bit is set when the rxipv4_hdrerr_octets counter reaches half the maximum value, and also when it reaches the maximum value. |
| 16      | V4GOCT     | Rx IPv4 (Good) Octets Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4GOCT bit is set when the rxipv4_gd_octets counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 13      | ICMPERRFRM | Rx ICMP Error Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.ICMPERRFRM bit is set when the rxicmp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.             |
| 12      | ICMPGFRM   | Rx ICMP (Good) Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.ICMPGFRM bit is set when the rxicmp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |

**Table 23-108:** EMAC\_IPC\_RXINT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                            |
|---------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11      | TCPERRFRM  | Rx TCP Error Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.TCPERRFRM bit is set when the rxtcp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.              |
| 10      | TCPGFRM    | Rx TCP (Good) Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.TCPGFRM bit is set when the rxtcp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 9       | UDPERFRM   | Rx IDP Error Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.UDPERFRM bit is set when the rxudp_err_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 8       | UDPGFRM    | Rx UDP (Good) Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.UDPGFRM bit is set when the rxudp_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.                |
| 7       | V6NOPAYFRM | Rx IPv6 No Payload Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6NOPAYFRM bit is set when the rxipv6_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 6       | V6HDERRFRM | Rx IPv6 Header Error Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6HDERRFRM bit is set when the rxipv6_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value. |
| 5       | V6GFRM     | Rx IPv6 (Good) Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V6GFRM bit is set when the rxipv6_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |
| 4       | V4UDSBLFRM | Rx IPv4 UDS Disable Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4UDSBLFRM bit is set when the rxipv4_udsbl_frms counter reaches half the maximum value, and also when it reaches the maximum value.   |

**Table 23-108:** EMAC\_IPC\_RXINT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                            |
|---------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | V4FRAGFRM  | Rx IPv4 Fragmented Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4FRAGFRM bit is set when the rxipv4_frag_frms counter reaches half the maximum value, and also when it reaches the maximum value.      |
| 2       | V4NOPAYFRM | Rx IPv4 No Payload Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4NOPAYFRM bit is set when the rxipv4_nopay_frms counter reaches half the maximum value, and also when it reaches the maximum value.    |
| 1       | V4HDERRFRM | Rx IPv4 Header Error Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4HDERRFRM bit is set when the rxipv4_hdrerr_frms counter reaches half the maximum value, and also when it reaches the maximum value. |
| 0       | V4GFRM     | Rx IPv4 (Good) Frames Count Half/Full Interrupt.<br>The EMAC_IPC_RXINT.V4GFRM bit is set when the rxipv4_gd_frms counter reaches half the maximum value, and also when it reaches the maximum value.               |

## Rx IPv4 Datagrams (Good) Register

The EMAC\_RXIPV4\_GD\_FRM register contains a count of the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload.

**EMAC\_RXIPV4\_GD\_FRM: Rx IPv4 Datagrams (Good) Register - R/NW**

Reset = 0x0000 0000

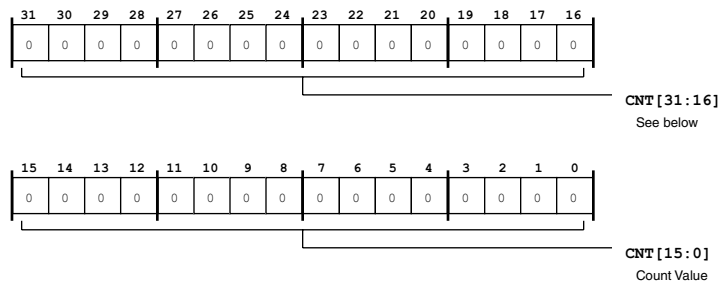


**Figure 23-85:** EMAC\_RXIPV4\_GD\_FRM Register Diagram

Table 23-109: EMAC\_RXIPV4\_GD\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx IPv4 Datagrams Header Errors Register

The EMAC\_RXIPV4\_HDR\_ERR\_FRM register contains a count of the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors.

EMAC\_RXIPV4\_HDR\_ERR\_FRM: Rx IPv4 Datagrams Header Errors Register - R/NW

Reset = 0x0000 0000

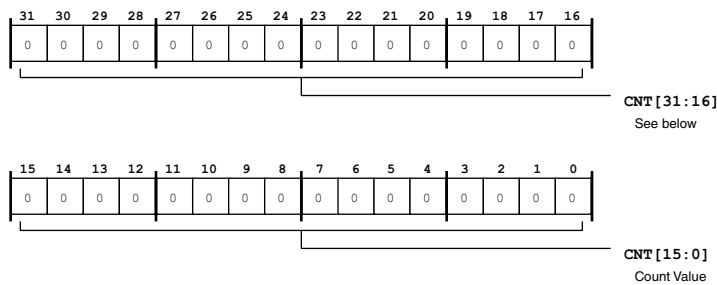


Figure 23-86: EMAC\_RXIPV4\_HDR\_ERR\_FRM Register Diagram

Table 23-110: EMAC\_RXIPV4\_HDR\_ERR\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx IPv4 Datagrams No Payload Frame Register

The EMAC\_RXIPV4\_NOPAY\_FRM register contains a count of the number of IPv4 datagram frames received that did not have a TCP, UDP, or ICMP payload processed by the Checksum engine.

EMAC\_RXIPV4\_NOPAY\_FRM: Rx IPv4 Datagrams No Payload Frame Register - R/NW

Reset = 0x0000 0000

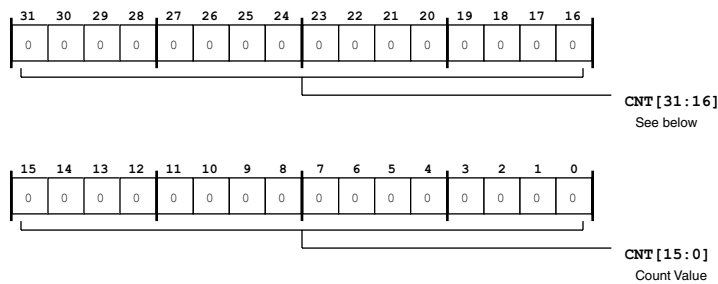


Figure 23-87: EMAC\_RXIPV4\_NOPAY\_FRM Register Diagram

Table 23-111: EMAC\_RXIPV4\_NOPAY\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx IPv4 Datagrams Fragmented Frames Register

The EMAC\_RXIPV4\_FRAG\_FRM register contains a count of the number of good IPv4 datagrams with fragmentation.

EMAC\_RXIPV4\_FRAG\_FRM: Rx IPv4 Datagrams Fragmented Frames Register - R/NW

Reset = 0x0000 0000

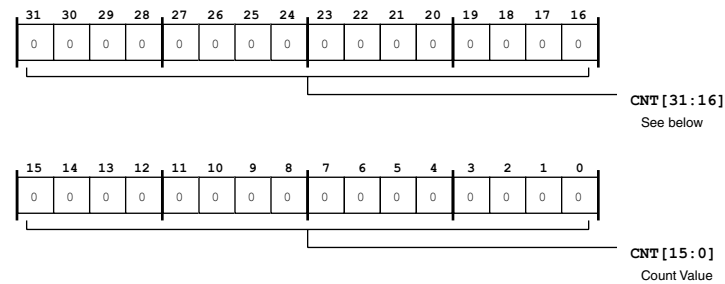


Figure 23-88: EMAC\_RXIPV4\_FRAG\_FRM Register Diagram

Table 23-112: EMAC\_RXIPV4\_FRAG\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv4 UDP Disabled Frames Register

The EMAC\_RXIPV4\_UDSBL\_FRM register contains a count of the number of good IPv4 datagrams received that had a UDP payload with checksum disabled.

EMAC\_RXIPV4\_UDSBL\_FRM: Rx IPv4 UDP Disabled Frames Register - R/NW

Reset = 0x0000 0000

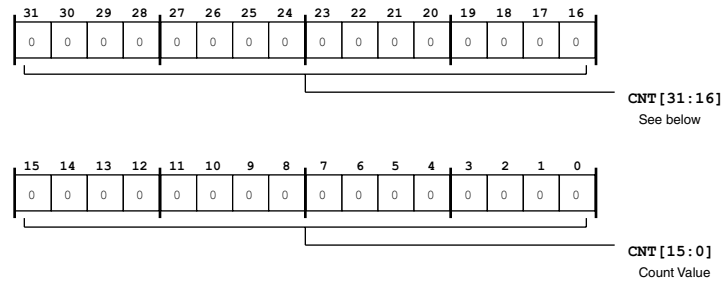


Figure 23-89: EMAC\_RXIPV4\_UDSBL\_FRM Register Diagram

Table 23-113: EMAC\_RXIPV4\_UDSBL\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv6 Datagrams Good Frames Register

The EMAC\_RXIPV6\_GD\_FRM register contains a count of the number of good IPv6 datagrams received with TCP, UDP, or ICMP payloads.

EMAC\_RXIPV6\_GD\_FRM: Rx IPv6 Datagrams Good Frames Register - R/NW

Reset = 0x0000 0000

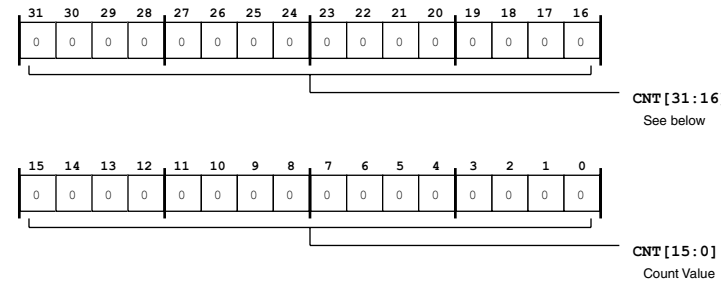


Figure 23-90: EMAC\_RXIPV6\_GD\_FRM Register Diagram

**Table 23-114:** EMAC\_RXIPV6\_GD\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv6 Datagrams Header Error Frames Register

The EMAC\_RXIPV6\_HDR\_ERR\_FRM register contains a count of the number of IPv6 datagrams received with header errors (length or version mismatch).

**EMAC\_RXIPV6\_HDR\_ERR\_FRM: Rx IPv6 Datagrams Header Error Frames Register - R/NW**

Reset = 0x0000 0000

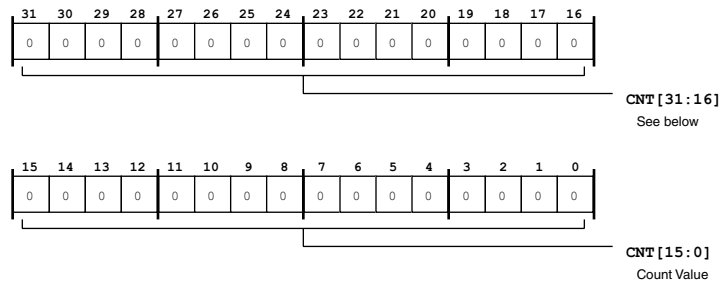


**Figure 23-91:** EMAC\_RXIPV6\_HDR\_ERR\_FRM Register Diagram

**Table 23-115:** EMAC\_RXIPV6\_HDR\_ERR\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv6 Datagrams No Payload Frames Register

The EMAC\_RXIPV6\_NOPAY\_FRM register contains a count of the number of IPv6 datagram frames received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

EMAC\_RXIPV6\_NOPAY\_FRM: Rx IPv6 Datagrams No Payload Frames Register - R/NW

Reset = 0x0000 0000

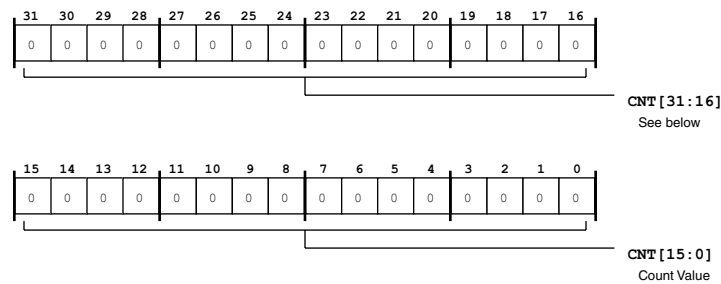


Figure 23-92: EMAC\_RXIPV6\_NOPAY\_FRM Register Diagram

Table 23-116: EMAC\_RXIPV6\_NOPAY\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx UDP Good Frames Register

The EMAC\_RXUDP\_GD\_FRM register contains a count of the number of good IP datagrams with a good UDP payload. This counter is not updated when the rxipv4\_udtbl\_frms counter is incremented.

EMAC\_RXUDP\_GD\_FRM: Rx UDP Good Frames Register - R/NW

Reset = 0x0000 0000

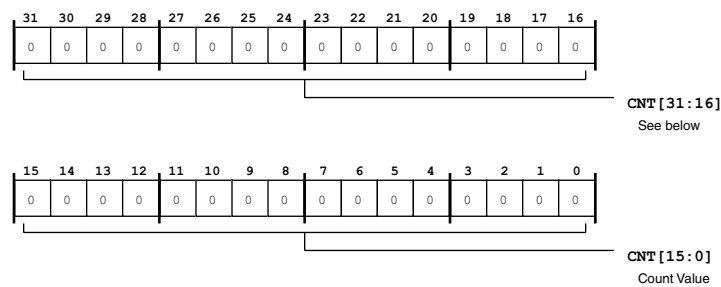


Figure 23-93: EMAC\_RXUDP\_GD\_FRM Register Diagram

Table 23-117: EMAC\_RXUDP\_GD\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |



## Rx UDP Error Frames Register

The EMAC\_RXUDP\_ERR\_FRM register contains a count of the number of good IP datagrams whose UDP payload has a checksum error.

EMAC\_RXUDP\_ERR\_FRM: Rx UDP Error Frames Register - R/NW

Reset = 0x0000 0000

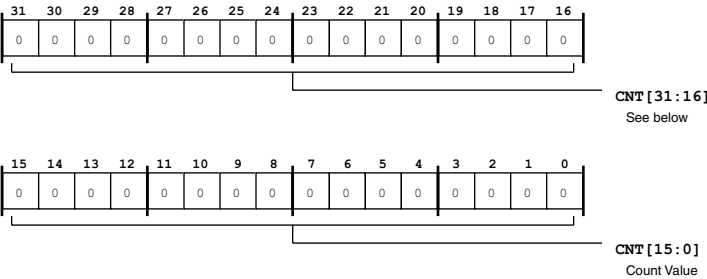


Figure 23-94: EMAC\_RXUDP\_ERR\_FRM Register Diagram

Table 23-118: EMAC\_RXUDP\_ERR\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx TCP Good Frames Register

The EMAC\_RXTCP\_GD\_FRM register contains a count of the number of good IP datagrams with a good TCP payload.

EMAC\_RXTCP\_GD\_FRM: Rx TCP Good Frames Register - R/NW

Reset = 0x0000 0000

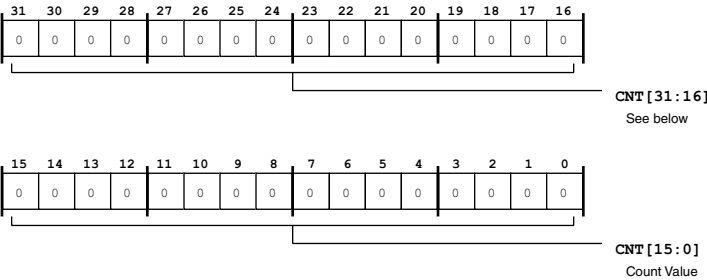


Figure 23-95: EMAC\_RXTCP\_GD\_FRM Register Diagram

Table 23-119: EMAC\_RXTCP\_GD\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx TCP Error Frames Register

The EMAC\_RXTCP\_ERR\_FRM register contains a count of the number of good IP datagrams whose TCP payload has a checksum error.

EMAC\_RXTCP\_ERR\_FRM: Rx TCP Error Frames Register - R/NW

Reset = 0x0000 0000

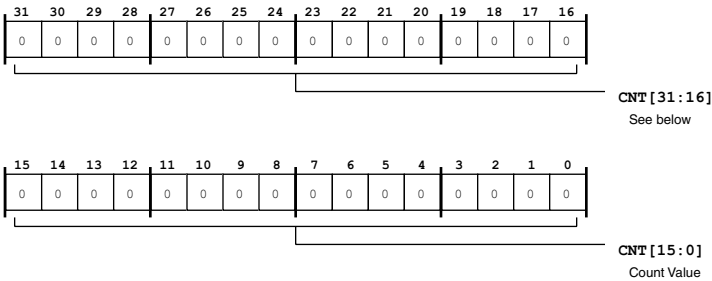


Figure 23-96: EMAC\_RXTCP\_ERR\_FRM Register Diagram

Table 23-120: EMAC\_RXTCP\_ERR\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx ICMP Good Frames Register

The EMAC\_RXICMP\_GD\_FRM register contains a count of the number of good IP datagrams with a good ICMP payload.

EMAC\_RXICMP\_GD\_FRM: Rx ICMP Good Frames Register - R/NW

Reset = 0x0000 0000

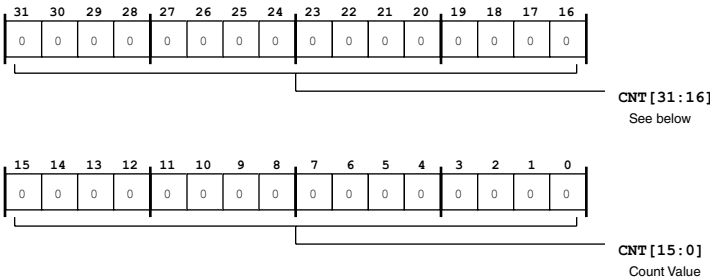


Figure 23-97: EMAC\_RXICMP\_GD\_FRM Register Diagram

Table 23-121: EMAC\_RXICMP\_GD\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx ICMP Error Frames Register

The EMAC\_RXICMP\_ERR\_FRM register contains a count of the number of good IP datagrams whose ICMP payload has a checksum error.

EMAC\_RXICMP\_ERR\_FRM: Rx ICMP Error Frames Register - R/NW

Reset = 0x0000 0000

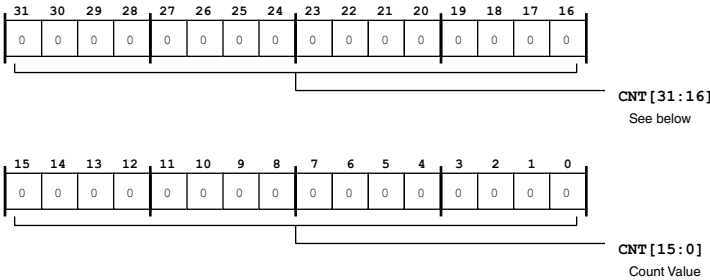


Figure 23-98: EMAC\_RXICMP\_ERR\_FRM Register Diagram

Table 23-122: EMAC\_RXICMP\_ERR\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv4 Datagrams Good Octets Register

The EMAC\_RXIPV4\_GD\_OCT register contains a count of the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data.

EMAC\_RXIPV4\_GD\_OCT: Rx IPv4 Datagrams Good Octets Register - R/NW

Reset = 0x0000 0000

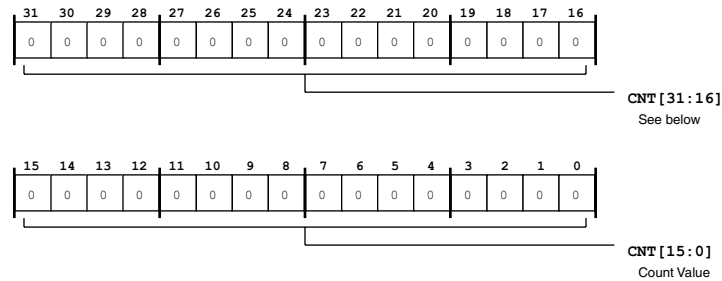


Figure 23-99: EMAC\_RXIPV4\_GD\_OCT Register Diagram

Table 23-123: EMAC\_RXIPV4\_GD\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv4 Datagrams Header Errors Register

The EMAC\_RXIPV4\_HDR\_ERR\_OCT register contains a count of the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter.

EMAC\_RXIPV4\_HDR\_ERR\_OCT: Rx IPv4 Datagrams Header Errors Register - R/NW

Reset = 0x0000 0000

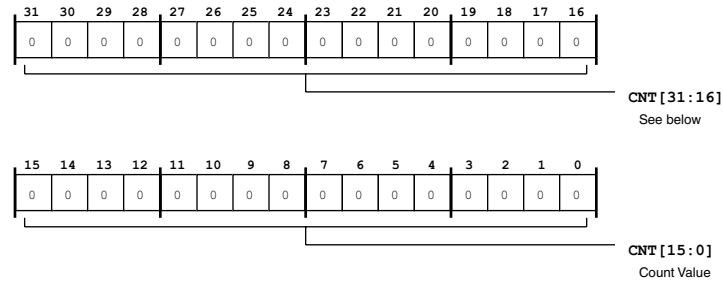


Figure 23-100: EMAC\_RXIPV4\_HDR\_ERR\_OCT Register Diagram

**Table 23-124:** EMAC\_RXIPV4\_HDR\_ERR\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv4 Datagrams No Payload Octets Register

The EMAC\_RXIPV4\_NOPAY\_OCT register contains a count of the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv4 headers Length field is used to update this counter.

**EMAC\_RXIPV4\_NOPAY\_OCT: Rx IPv4 Datagrams No Payload Octets Register - R/NW**

Reset = 0x0000 0000

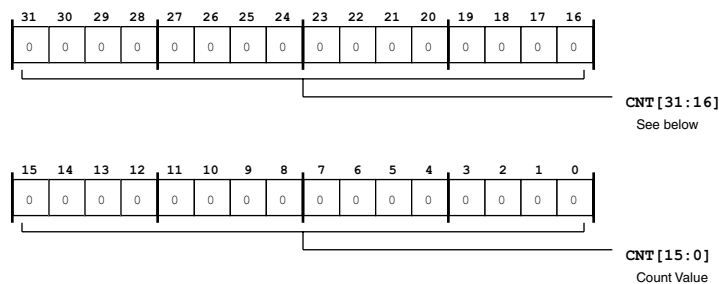


**Figure 23-101:** EMAC\_RXIPV4\_NOPAY\_OCT Register Diagram

**Table 23-125:** EMAC\_RXIPV4\_NOPAY\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv4 Datagrams Fragmented Octets Register

The EMAC\_RXIPV4\_FRAG\_OCT register contains a count of the number of bytes received in fragmented IPv4 datagrams. The value in the IPv4 headers Length field is used to update this counter.

EMAC\_RXIPV4\_FRAG\_OCT: Rx IPv4 Datagrams Fragmented Octets Register - R/NW

Reset = 0x0000 0000

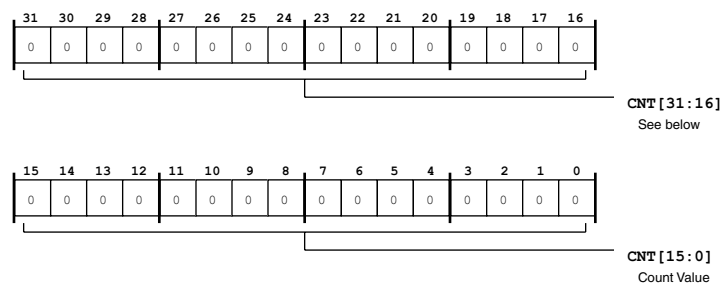


Figure 23-102: EMAC\_RXIPV4\_FRAG\_OCT Register Diagram

Table 23-126: EMAC\_RXIPV4\_FRAG\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx IPv4 UDP Disabled Octets Register

The EMAC\_RXIPV4\_UDSBL\_OCT register contains a count of the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes.

EMAC\_RXIPV4\_UDSBL\_OCT: Rx IPv4 UDP Disabled Octets Register - R/NW

Reset = 0x0000 0000

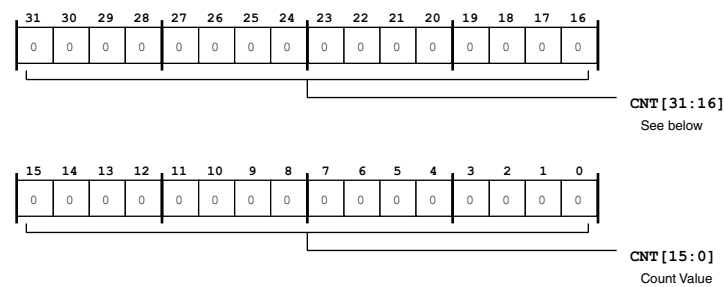


Figure 23-103: EMAC\_RXIPV4\_UDSBL\_OCT Register Diagram

Table 23-127: EMAC\_RXIPV4\_UDSBL\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv6 Good Octets Register

The EMAC\_RXIPV6\_GD\_OCT register contains a count of the number of bytes received in good IPv6 data-grams encapsulating TCP, UDP or ICMPv6 data

EMAC\_RXIPV6\_GD\_OCT: Rx IPv6 Good Octets Register - R/NW

Reset = 0x0000 0000

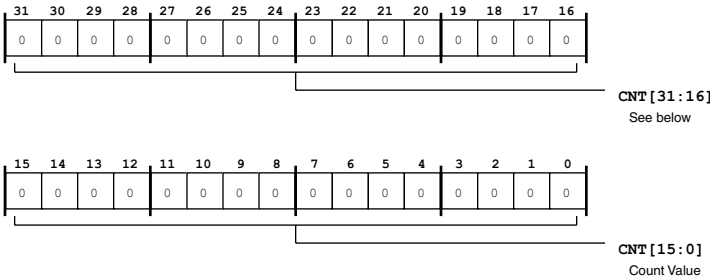


Figure 23-104: EMAC\_RXIPV6\_GD\_OCT Register Diagram

Table 23-128: EMAC\_RXIPV6\_GD\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx IPv6 Header Errors Register

The EMAC\_RXIPV6\_HDR\_ERR\_OCT register contains a count of the number of bytes received in IPv6 data-grams with header errors (length, version mismatch). The value in the IPv6 headers Length field is used to update this counter.

EMAC\_RXIPV6\_HDR\_ERR\_OCT: Rx IPv6 Header Errors Register - R/NW

Reset = 0x0000 0000

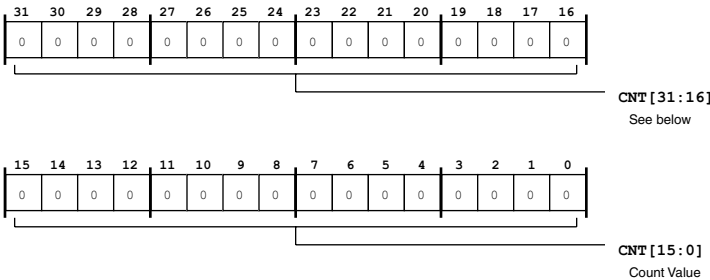


Figure 23-105: EMAC\_RXIPV6\_HDR\_ERR\_OCT Register Diagram

Table 23-129: EMAC\_RXIPV6\_HDR\_ERR\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx IPv6 No Payload Octets Register

The EMAC\_RXIPV6\_NOPAY\_OCT register contains a count of the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the IPv6 headers Length field is used to update this counter.

EMAC\_RXIPV6\_NOPAY\_OCT: Rx IPv6 No Payload Octets Register - R/NW

Reset = 0x0000 0000

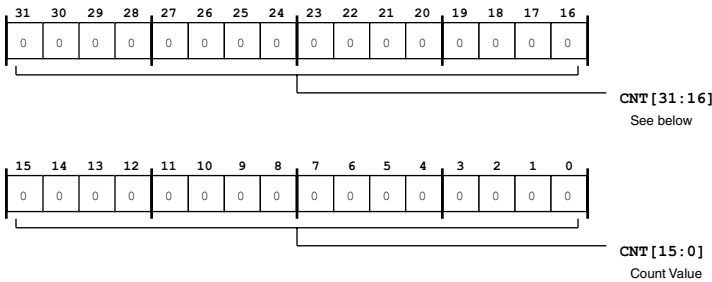


Figure 23-106: EMAC\_RXIPV6\_NOPAY\_OCT Register Diagram

Table 23-130: EMAC\_RXIPV6\_NOPAY\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx UDP Good Octets Register

The EMAC\_RXUDP\_GD\_OCT register contains a count of the number of bytes received in a good UDP segment. This counter (and the counters below) does not count IP header bytes.



EMAC\_RXUDP\_GD\_OCT: Rx UDP Good Octets Register - R/NW

Reset = 0x0000 0000

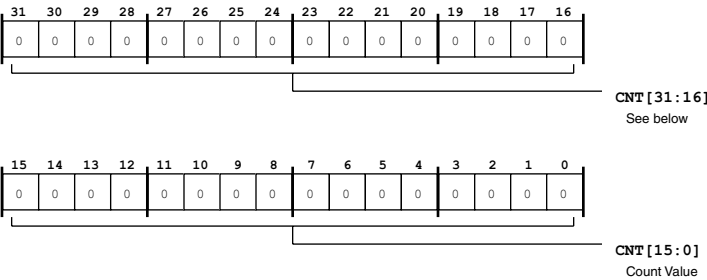


Figure 23-107: EMAC\_RXUDP\_GD\_OCT Register Diagram

Table 23-131: EMAC\_RXUDP\_GD\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx UDP Error Octets Register

The EMAC\_RXUDP\_ERR\_OCT register contains a count of the number of bytes received in a UDP segment that had checksum errors.

EMAC\_RXUDP\_ERR\_OCT: Rx UDP Error Octets Register - R/NW

Reset = 0x0000 0000

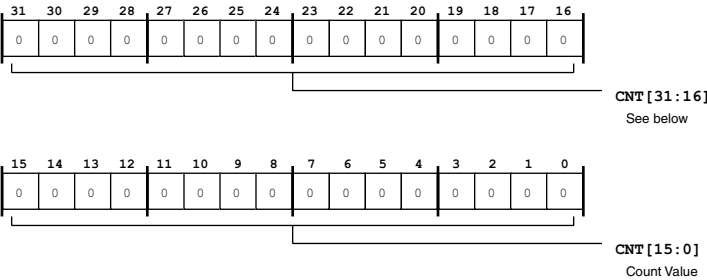


Figure 23-108: EMAC\_RXUDP\_ERR\_OCT Register Diagram

Table 23-132: EMAC\_RXUDP\_ERR\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx TCP Good Octets Register

The EMAC\_RXTCP\_GD\_OCT register contains a count of the number of bytes received in a good TCP segment.

EMAC\_RXTCP\_GD\_OCT: Rx TCP Good Octets Register - R/NW

Reset = 0x0000 0000

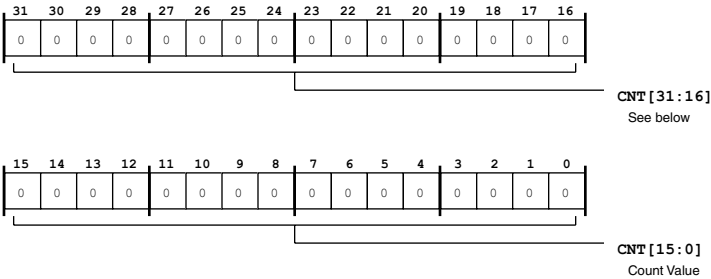


Figure 23-109: EMAC\_RXTCP\_GD\_OCT Register Diagram

Table 23-133: EMAC\_RXTCP\_GD\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

## Rx TCP Error Octets Register

The EMAC\_RXTCP\_ERR\_OCT register contains a count of the number of bytes received in a TCP segment with checksum errors.

EMAC\_RXTCP\_ERR\_OCT: Rx TCP Error Octets Register - R/NW

Reset = 0x0000 0000

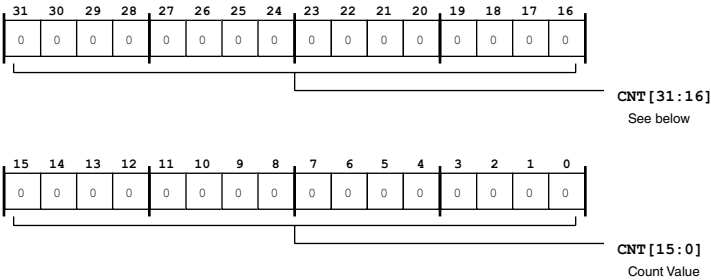


Figure 23-110: EMAC\_RXTCP\_ERR\_OCT Register Diagram

Table 23-134: EMAC\_RXTCP\_ERR\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx ICMP Good Octets Register

The EMAC\_RXICMP\_GD\_OCT register contains a count of the Number of bytes received in a good ICMP segment.

EMAC\_RXICMP\_GD\_OCT: Rx ICMP Good Octets Register - R/NW

Reset = 0x0000 0000

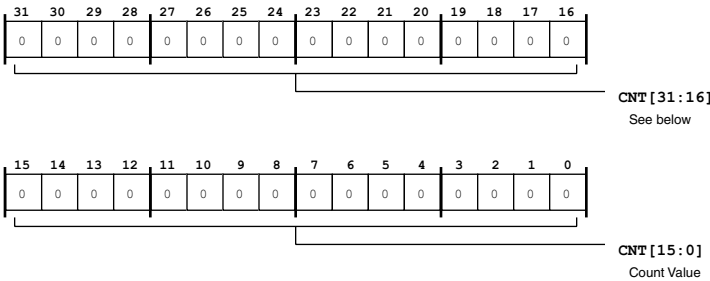


Figure 23-111: EMAC\_RXICMP\_GD\_OCT Register Diagram

Table 23-135: EMAC\_RXICMP\_GD\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Rx ICMP Error Octets Register

The EMAC\_RXICMP\_ERR\_OCT register contains a count of the number of bytes received in an ICMP segment with checksum errors.

EMAC\_RXICMP\_ERR\_OCT: Rx ICMP Error Octets Register - R/NW

Reset = 0x0000 0000

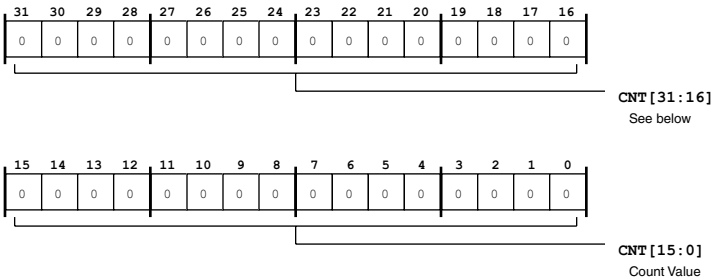


Figure 23-112: EMAC\_RXICMP\_ERR\_OCT Register Diagram

Table 23-136: EMAC\_RXICMP\_ERR\_OCT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | CNT      | Count Value.            |

Time Stamp Control Register

The EMAC\_TM\_CTL register controls time stamp generation and update. The EMAC\_TM\_CTL.SNAPTYPSEL, EMAC\_TM\_CTL.TSMSTRENA, and EMAC\_TM\_CTL.TSEVNTENA bits work together to decide the set of PTP packet types for which snapshot needs to be taken. (Encoding shown in table.)

Table 23-137:

| SNAPTYPSEL() | TSMSTRENA | TSEVNTENA | Messages for which snapshot is taken                                                   |
|--------------|-----------|-----------|----------------------------------------------------------------------------------------|
| 00           | X         | 0         | SYNC, Follow_Up, Delay_Req, Delay_Resp                                                 |
| 00           | 0         | 1         | SYNC                                                                                   |
| 00           | 1         | 1         | Delay_Req                                                                              |
| 01           | X         | 0         | SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up |
| 01           | 0         | 1         | SYNC, Pdelay_Req, Pdelay_Resp                                                          |
| 01           | 1         | 1         | Delay_Req, Pdelay_Req, Pdelay_Resp                                                     |
| 10           | X         | X         | SYNC, Delay_Req                                                                        |

**Table 23-137:** (Continued)

| SNAPTYPSEL() | TSMSTRENA | TSEVNTENA | Messages for which snapshot is taken |
|--------------|-----------|-----------|--------------------------------------|
| 11           | X         | X         | Pdelay_Req, Pdelay_Resp              |

**EMAC\_TM\_CTL: Time Stamp Control Register - R/W**

Reset = 0x0000 2000

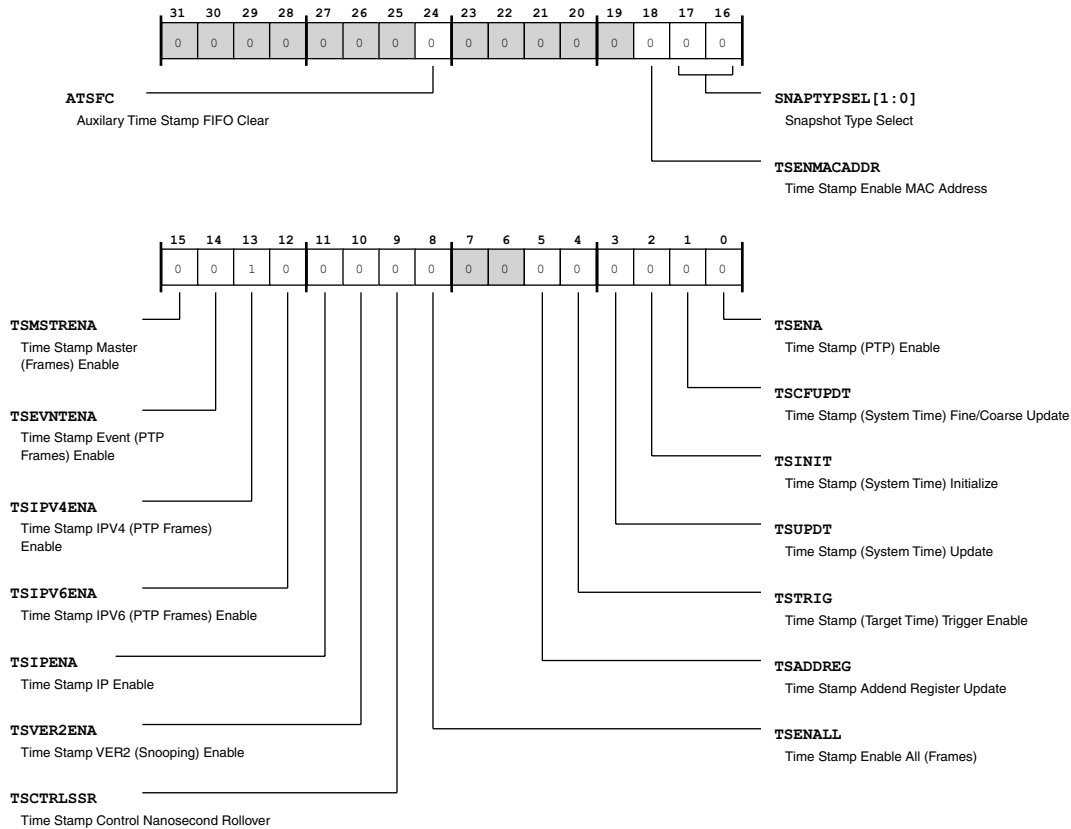


**Figure 23-113:** EMAC\_TM\_CTL Register Diagram

**Table 23-138:** EMAC\_TM\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24      | ATSF     | Auxiliary Time Stamp FIFO Clear.<br>The EMAC_TM_CTL.ATSF bit, when set, resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is cleared, auxiliary snapshots gets stored in the FIFO. |

**Table 23-138:** EMAC\_TM\_CTL Register Fields (Continued)

| Bit No. | Bit Name    | Description/Enumeration                                                                                                                                                                                                                                                                       |
|---------|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | TSENMACADDR | Time Stamp Enable MAC Address.<br>The EMAC_TM_CTL.TSENMACADDR bit, when set, uses the DA MAC address (that matches the MAC Address register) to filter the PTP frames when PTP is sent directly over Ethernet.                                                                                |
|         |             | 0      Disable PTP MAC address filter                                                                                                                                                                                                                                                         |
|         |             | 1      Enable PTP MAC address filter                                                                                                                                                                                                                                                          |
| 17:16   | SNAPTYPSEL  | Snapshot Type Select.<br>The EMAC_TM_CTL.SNAPTYPSEL bits along with bit 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken. (See the table in the EMAC_TM_CTL register description.)                                                                           |
| 15      | TSMSTRENA   | Time Stamp Master (Frames) Enable.<br>The EMAC_TM_CTL.TSMSTRENA bit, when set, takes the snapshot for messages relevant to master node only else snapshot is taken for PTP messages relevant to slave node.                                                                                   |
|         |             | 0      Enable Snapshot for Slave Messages                                                                                                                                                                                                                                                     |
|         |             | 1      Enable Snapshot for Master Messages                                                                                                                                                                                                                                                    |
| 14      | TSEVNTENA   | Time Stamp Event (PTP Frames) Enable.<br>The EMAC_TM_CTL.TSEVNTENA bit, when set, takes the time stamp snapshot for PTP event messages only (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all PTP messages except Announce, Management, and Signaling. |
|         |             | 0      Enable Time Stamp for All Messages                                                                                                                                                                                                                                                     |
|         |             | 1      Enable Time Stamp for Event Messages Only                                                                                                                                                                                                                                              |

**Table 23-138:** EMAC\_TM\_CTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                     |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | TSIPV4ENA | Time Stamp IPV4 (PTP Frames) Enable.<br>The EMAC_TM_CTL.TSIPV4ENA bit, when set, directs the EMAC receiver to process the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default. |
|         |           | 0      Disable Time Stamp for PTP Over IPv4 Frames                                                                                                                                                                                                                                          |
|         |           | 1      Enable Time Stamp for PTP Over IPv4 Frames                                                                                                                                                                                                                                           |
| 12      | TSIPV6ENA | Time Stamp IPV6 (PTP Frames) Enable.<br>The EMAC_TM_CTL.TSIPV6ENA bit, when set, directs the EMAC receiver to process PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets.                                 |
|         |           | 0      Disable Time Stamp for PTP Over IPv6 frames                                                                                                                                                                                                                                          |
|         |           | 1      Enable Time Stamp for PTP Over IPv6 Frames                                                                                                                                                                                                                                           |
| 11      | TSIPENA   | Time Stamp IP Enable.<br>The EMAC_TM_CTL.TSIPENA bit, when set, directs the EMAC receiver to process the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores PTP over Ethernet packets.                                                       |
|         |           | 0      Disable PTP Over Ethernet Frames                                                                                                                                                                                                                                                     |
|         |           | 1      Enable PTP Over Ethernet Frames                                                                                                                                                                                                                                                      |
| 10      | TSVER2ENA | Time Stamp VER2 (Snooping) Enable.<br>The EMAC_TM_CTL.TSVER2ENA bit, when set, processes the PTP packets using the 1588 version 2 format (enables PTP packet snooping for VER2) else processed using the version 1 format.                                                                  |
|         |           | 0      Disable packet snooping for V2 frames                                                                                                                                                                                                                                                |
|         |           | 1      Enable packet snooping for V2 frames                                                                                                                                                                                                                                                 |

**Table 23-138:** EMAC\_TM\_CTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | TSCTRLSSR | Time Stamp Control Nanosecond Rollover.<br>The EMAC_TM_CTL.TSCTRLSSR bit, when set, rolls over the Time Stamp nanoseconds register after 0x3B9A_C9FF value ( $10^9-1$ ) and increments the time stamp (low) seconds. When reset, the roll over value of nanosecond register is 0x7FFF_FFFF. The nanosecond increment has to be programmed correctly depending on the PTP reference clock frequency and this bit value. |
|         |           | 0      Roll Over Nanosecond After 0x7FFFFFFF                                                                                                                                                                                                                                                                                                                                                                           |
|         |           | 1      Roll Over Nanosecond After 0x3B9AC9FF                                                                                                                                                                                                                                                                                                                                                                           |
| 8       | TSENALL   | Time Stamp Enable All (Frames).<br>The EMAC_TM_CTL.TSENALL bit, when set, enables the time stamp snapshot for all frames received by the core.                                                                                                                                                                                                                                                                         |
|         |           | 0      Disable timestamp for all frames                                                                                                                                                                                                                                                                                                                                                                                |
|         |           | 1      Enable timestamp for all frames                                                                                                                                                                                                                                                                                                                                                                                 |
| 5       | TSADDREG  | Time Stamp Addend Register Update.<br>The EMAC_TM_CTL.TSADDREG bit, when set, updates the contents of the Time Stamp Addend register in the PTP block for fine correction. This bit is cleared when the update is completed. This bit should be zero before setting it.                                                                                                                                                |
| 4       | TSTRIG    | Time Stamp (Target Time) Trigger Enable.<br>The EMAC_TM_CTL.TSTRIG bit, when set, generates the time stamp interrupt when the System Time becomes greater than the value written in Target Time register. This bit is reset after the generation of the Time Stamp Trigger Interrupt.                                                                                                                                  |
|         |           | 1      Interrupt (TS) if system time is greater than target time register                                                                                                                                                                                                                                                                                                                                              |



**Table 23-138:** EMAC\_TM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | TSUPDT   | Time Stamp (System Time) Update.<br>The EMAC_TM_CTL.TSUPDT bit, when set, updates (adds/subtracts) the system time with the value specified in System Time - Seconds Update Register and System Time - Nanoseconds Update Register. This register bit should be read zero before updating it. This bit is reset once the update is completed in hardware. The "Time Stamp Higher Word" register is not updated.          |
|         |          | 1 System time updated with Time stamp register values                                                                                                                                                                                                                                                                                                                                                                    |
| 2       | TSINIT   | Time Stamp (System Time) Initialize.<br>The EMAC_TM_CTL.TSINIT bit, when set, initializes (over-writes) the system time with the value specified in the System Time - Seconds Update Register and System Time - Nanoseconds Update Register. This register bit should be read zero before updating it. This bit is reset once the initialize is complete. The "Time Stamp Higher Word" register can only be initialized. |
|         |          | 1 System time initialized with Time stamp register values                                                                                                                                                                                                                                                                                                                                                                |
| 1       | TSCFUPDT | Time Stamp (System Time) Fine/Coarse Update.<br>The EMAC_TM_CTL.TSCFUPDT bit, when set, indicates that the system times update to be done using fine correction method. When reset, it indicates the system time stamp correction to be done using Coarse method.                                                                                                                                                        |
|         |          | 0 Use Coarse Correction Method for System Time Update                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Use Fine Correction Method for System Time Update                                                                                                                                                                                                                                                                                                                                                                      |

Table 23-138: EMAC\_TM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | TSENA    | Time Stamp (PTP) Enable.<br>The EMAC_TM_CTL.TSENA bit, when set, enables PTP module for timestamping transmitted and received frames. It also enables System Time which will be used for timestamping the frames. User should initialize the System Time after setting this bit. |
|         |          | 0      Disable PTP Module                                                                                                                                                                                                                                                        |
|         |          | 1      Enable PTP Module                                                                                                                                                                                                                                                         |

Time Stamp Sub Second Increment Register

The EMAC\_TM\_SUBSEC register contains the value by which the sub second is incremented.

EMAC\_TM\_SUBSEC: Time Stamp Sub Second Increment Register - R/W

Reset = 0x0000 0000

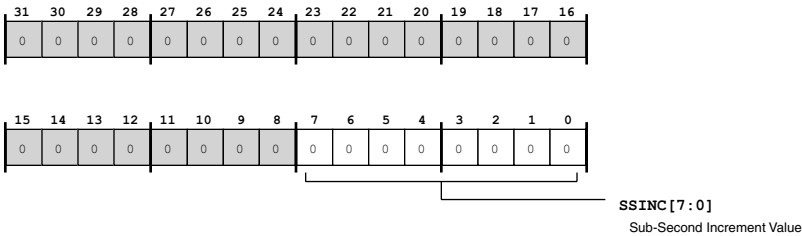


Figure 23-114: EMAC\_TM\_SUBSEC Register Diagram

Table 23-139: EMAC\_TM\_SUBSEC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | SSINC    | Sub-Second Increment Value.<br>The value in the EMAC_TM_SUBSEC.SSINC bits is accumulated every PTP clock cycle with the contents of the nanosecond register. For example, when PTP clock is 50 MHz (period is 20 ns), the processor should program 20 (0x14) when the Time Stamp Nanoseconds register has an accuracy of 1 ns (TSCTRLSSR bit is set). When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465ns. In this case, the processor should program a value of 43 (0x2B) that is derived by 20ns/0.465. |

## Time Stamp Low Seconds Register

The EMAC\_TM\_SEC register contains the lower 32 bits of the seconds field of the system time.

EMAC\_TM\_SEC: Time Stamp Low Seconds Register - R/W

Reset = 0x0000 0000

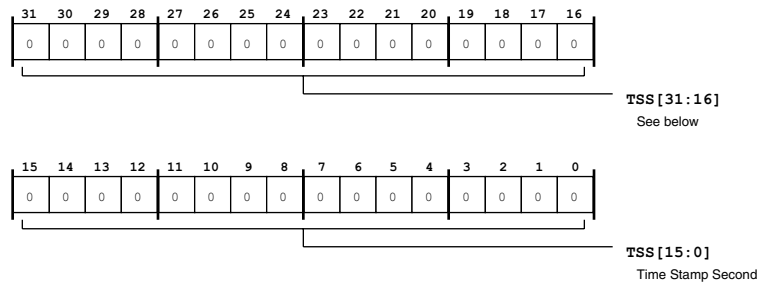


Figure 23-115: EMAC\_TM\_SEC Register Diagram

Table 23-140: EMAC\_TM\_SEC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TSS      | Time Stamp Second.<br>The value in the EMAC_TM_SEC.TSS bit field indicates the current value in seconds of the System Time maintained by the core. |

## Time Stamp Nano Seconds Register

The EMAC\_TM\_NSEC register contains the nano-seconds field of the system time.

EMAC\_TM\_NSEC: Time Stamp Nano Seconds Register - R/W

Reset = 0x0000 0000

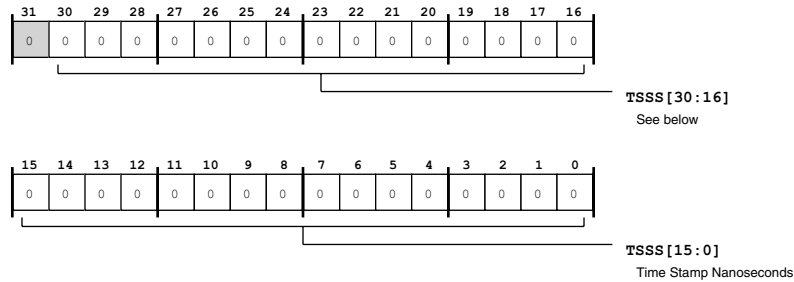


Figure 23-116: EMAC\_TM\_NSEC Register Diagram

Table 23-141: EMAC\_TM\_NSEC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30:0    | TSSS     | Time Stamp Nanoseconds.<br>The value in the EMAC_TM_NSEC.TSSS bit field has the nanosecond representation of time, with an accuracy of 0.46 nano-second. (When TSCTRLSSR is set, each bit represents 1 ns and the maximum value will be 0x3B9A_C9FF, after which it rolls-over to zero). |

Time Stamp Seconds Update Register

The EMAC\_TM\_SECUPDT register contains the low 32 bits to be added to, subtracted from, or written to the seconds field of the system time.

EMAC\_TM\_SECUPDT: Time Stamp Seconds Update Register - R/W

Reset = 0x0000 0000

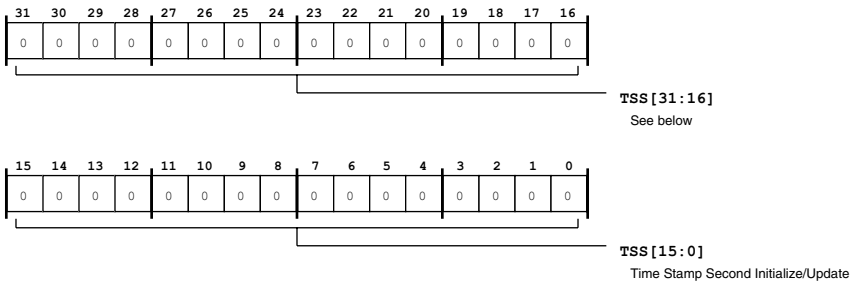


Figure 23-117: EMAC\_TM\_SECUPDT Register Diagram

Table 23-142: EMAC\_TM\_SECUPDT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TSS      | Time Stamp Second Initialize/Update.<br>The value in the EMAC_TM_SECUPDT.TSS bit field indicates the time, in seconds, to be initialized or added to or subtracted from the system time seconds. |

Time Stamp Nano Seconds Update Register

The EMAC\_TM\_NSECUPDT register contains the low 32 bits to be added to, subtracted from, or written to the nano-seconds field of the system time.

EMAC\_TM\_NSECUPDT: Time Stamp Nano Seconds Update Register - R/W

Reset = 0x0000 0000

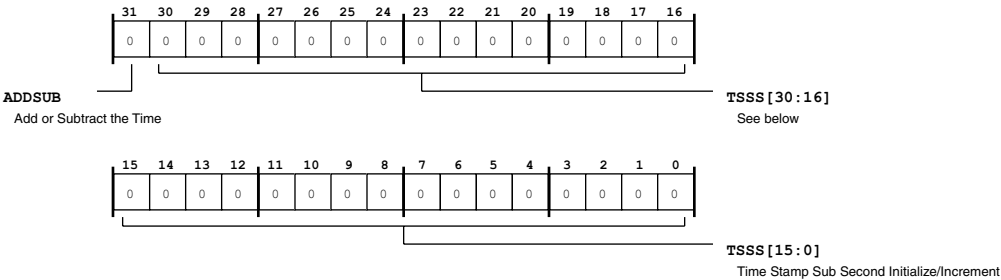


Figure 23-118: EMAC\_TM\_NSECUPDT Register Diagram

Table 23-143: EMAC\_TM\_NSECUPDT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | ADDSUB   | Add or Subtract the Time.<br>The EMAC_TM_NSECUPDT .ADDSUB bit, when set, subtracts the time value with the contents of the update registers. When this bit is reset, the time value is added with the contents of the update registers. |
| 30:0    | TSSS     | Time Stamp Sub Second Initialize/Increment.<br>The value in the EMAC_TM_NSECUPDT .TSSS bit field indicates the time, in nanoseconds, to be initialized or added to or subtracted from the system time nanoseconds.                      |

Time Stamp Addend Register

The EMAC\_TM\_ADDEND register lets software adjust the clock frequency linearly to match the master clock frequency.

EMAC\_TM\_ADDEND: Time Stamp Addend Register - R/W

Reset = 0x0000 0000

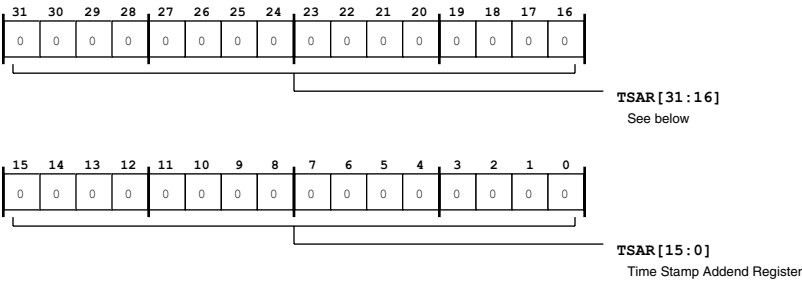


Figure 23-119: EMAC\_TM\_ADDEND Register Diagram

Table 23-144: EMAC\_TM\_ADDEND Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TSAR     | Time Stamp Addend Register.<br>The EMAC_TM_ADDEND.TSAR bits indicate the 32-bit time value to be added to the Accumulator register to achieve time synchronization. |

Time Stamp Target Time Seconds Register

The EMAC\_TM\_TGTM register contains the high 32 bits of the target seconds field for comparison to the corresponding system time field.

EMAC\_TM\_TGTM: Time Stamp Target Time Seconds Register - R/W

Reset = 0x0000 0000

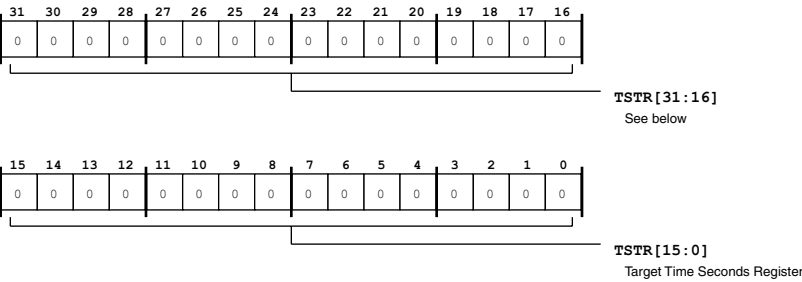


Figure 23-120: EMAC\_TM\_TGTM Register Diagram

**Table 23-145:** EMAC\_TM\_TGTM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TSTR     | Target Time Seconds Register.<br>The EMAC_TM_TGTM.TSTR bit field stores the time in seconds. When the time stamp value matches or exceeds both Target Time Stamp registers, then based on the bits 6:5 (TRGTMODSEL) of PPS Control Register, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). |

## Time Stamp Target Time Nano Seconds Register

The EMAC\_TM\_NTGMTM register contains the high 32 bits of the target nano-seconds field for comparison to the corresponding system time field.

**EMAC\_TM\_NTGMTM: Time Stamp Target Time Nano Seconds Register - R/W**

Reset = 0x0000 0000

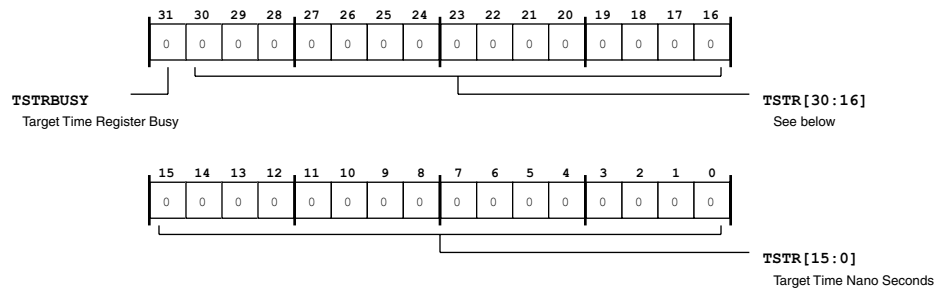


**Figure 23-121:** EMAC\_TM\_NTGMTM Register Diagram

**Table 23-146:** EMAC\_TM\_NTGMTM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | TSTRBUSY | Target Time Register Busy.<br>The EMAC_TM_NTGMTM.TSTRBUSY bit is set when Flexible PPS is enabled and the PPSCCTL field (bit 3:0) in PPS Control Register is programmed to 0001, 0010 or 0100. Programming the PPSCCTL field to 0001, 0010 or 0100, instructs the core to synchronize the Target Time Registers to the PTP clock domain. The EMAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain. The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. |

Table 23-146: EMAC\_TM\_NTGTM Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30:0    | TSTR     | Target Time Nano Seconds.<br>The EMAC_TM_NTGTM.TSTR bit field stores the time in (signed) nanoseconds. When the value of the time stamp matches the both Target Time Stamp registers, then based on the TRGTMODSEL field (bits 6:5) in PPS Control Register, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). This value should not exceed 0x3B9A_C9FF when TSCTRLSSR is set in the Timestamp control register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. |

Time Stamp High Second Register

The EMAC\_TM\_HISEC register contains the upper 32 bits of the seconds field of the system time.

EMAC\_TM\_HISEC: Time Stamp High Second Register - R/W

Reset = 0x0000 0000

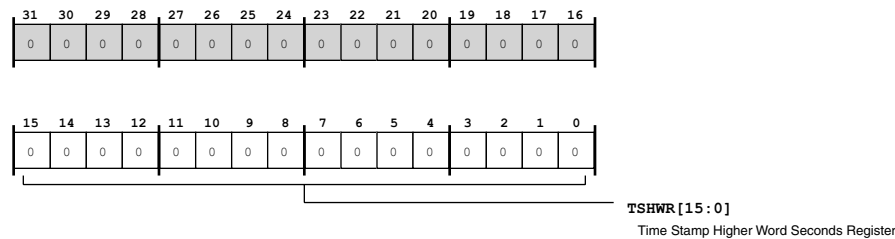


Figure 23-122: EMAC\_TM\_HISEC Register Diagram

Table 23-147: EMAC\_TM\_HISEC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | TSHWR    | Time Stamp Higher Word Seconds Register.<br>The EMAC_TM_HISEC.TSHWR bit field contains the most significant 16-bits of the time stamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the Time Stamp Low Seconds register. |



## Time Stamp Status Register

The EMAC\_TM\_STMPSTAT register contains the PTP status.

### EMAC\_TM\_STMPSTAT: Time Stamp Status Register - R/W

Reset = 0x0000 0000

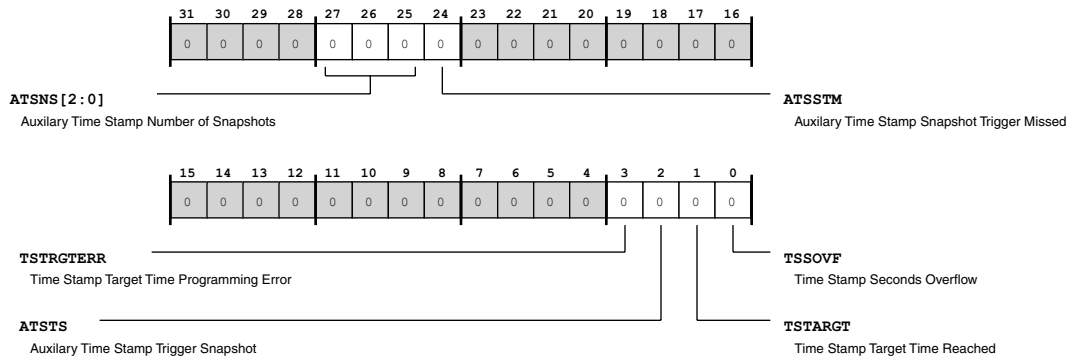


**Figure 23-123:** EMAC\_TM\_STMPSTAT Register Diagram

**Table 23-148:** EMAC\_TM\_STMPSTAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                        |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27:25   | ATSNS     | Auxiliary Time Stamp Number of Snapshots.<br>The EMAC_TM_STMPSTAT.ATSNS bits indicate the number of Snapshots available in the FIFO. A value of 4 (100) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 000) when the Auxiliary snapshot FIFO clear bit is set. |
| 24      | ATSSTM    | Auxiliary Time Stamp Snapshot Trigger Missed.<br>The EMAC_TM_STMPSTAT.ATSSTM bit is set when the Auxiliary time stamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot was not stored in the FIFO.                                                  |
| 3       | TSTRGTERR | Time Stamp Target Time Programming Error.<br>The EMAC_TM_STMPSTAT.TSTRGTERR bit is set when the target time, which is being programmed in Target Time Registers, has already elapsed. This bit is cleared when read by the application.                                                        |
| 2       | ATSTS     | Auxiliary Time Stamp Trigger Snapshot.<br>The EMAC_TM_STMPSTAT.ATSTS bit is set high when the auxiliary snapshot is written to the FIFO.                                                                                                                                                       |

**Table 23-148:** EMAC\_TM\_STMPSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | TSTARGET | Time Stamp Target Time Reached.<br>The EMAC_TM_STMPSTAT.TSTARGET bit, when set, indicates the value of system time has reached or passed the value specified in the Target Time Seconds and Nanoseconds registers |
| 0       | TSSOVF   | Time Stamp Seconds Overflow.<br>The EMAC_TM_STMPSTAT.TSSOVF bit, when set, indicates that the seconds value of the time stamp (when supporting PTP version 2 format) has overflowed beyond 32'hFFFF_FFFF.         |

## PPS Control Register

The EMAC\_TM\_PPSCCTL register controls the interval of PPS output.

When the EMAC\_TM\_PPSCCTL.PPSEN bit is disabled (=0, fixed PPS output), the EMAC\_TM\_PPSCCTL.PPSCCTL bits control the behavior of the PPS output signal. The default value of PPSCCTL is 0000 and the PPS output is 1 pulse every second. For other values of PPSCCTL, the PPS output becomes a generated clock. (See bit enumerations for frequencies.) In the binary rollover mode, the PPS output has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. This behavior is because of the non-linear toggling of the bits in the digital rollover mode in System Time - Nanoseconds Register.

When the EMAC\_TM\_PPSCCTL.PPSEN bit is enabled (=1, flexible PPS output), the EMAC\_TM\_PPSCCTL.PPSCCTL bits function as PPSCMD. (See bit enumerations for commands.) Programming these bits with a non-zero value instructs the core to initiate an event. After the command is transferred or synchronized to the PTP clock domain, these bits gets cleared automatically. Software should ensure that these bits are programmed only when they are "all-zero".

- EMAC\_TM\_PPSCCTL.PPSCCTL=0000 CMD=No Command (PPS output =1Hz)
- EMAC\_TM\_PPSCCTL.PPSCCTL=0001  
(CMD=START Single; BR=2kHz; DR=1kHz) This PPS command generates single pulse rising at start point defined in Target Time Registers and of duration defined in PPS Width Register. (Binary Rollover = 2 kHz; Digital Rollover = 1 kHz.)
- EMAC\_TM\_PPSCCTL.PPSCCTL=0010  
(CMD=START Pulse; BR=4kHz; DR=2kHz) This PPS command generates the train of pulses rising at the start point defined in the Target Time Registers and of duration defined in the PPS Width Register

and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands. (Binary Rollover = 4 kHz; Digital Rollover = 2 kHz.)

- EMAC\_TM\_PPSCTL.PPSCTL=0011

(CMD=Cancel START; BR=8kHz; DR=4kHz) This PPS command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time. (Binary Rollover = 4 kHz; Digital Rollover = 4 kHz.)

- EMAC\_TM\_PPSCTL.PPSCTL=0100

(CMD=STOP Pulse Time; BR=16kHz; DR=8kHz) This PPS command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses. (Binary Rollover = 16 kHz; Digital Rollover = 8 kHz.)

- EMAC\_TM\_PPSCTL.PPSCTL=0101

(CMD=STOP Pulse Now) This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010).

- EMAC\_TM\_PPSCTL.PPSCTL=0110

(CMD=Cancel STOP Pulse) This PPS command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

All values not shown in the bit enumerations are reserved.

#### EMAC\_TM\_PPSCTL: PPS Control Register - R/W

Reset = 0x0000 0000

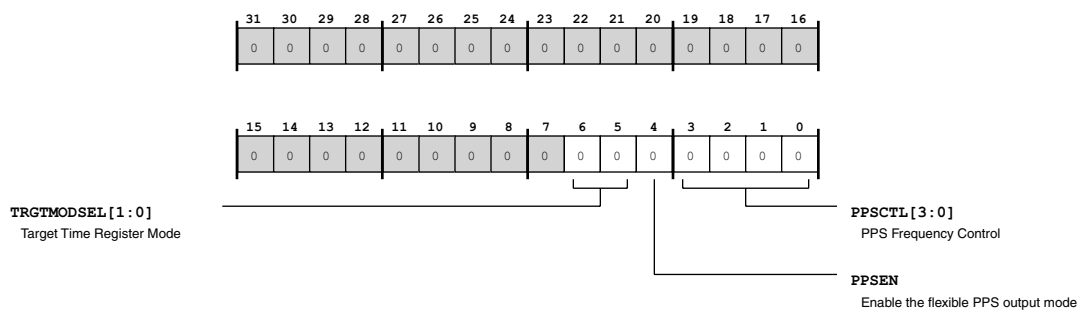


**Figure 23-124:** EMAC\_TM\_PPSCTL Register Diagram

**Table 23-149:** EMAC\_TM\_PPCTL Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                      |
|---------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:5     | TRGTMODSEL | Target Time Register Mode.<br>The EMAC_TM_PPCTL.TRGTMODSEL bits select the target time register mode.                                                                                                                                        |
|         |            | 0<br>Interrupt Only<br>The Target Time registers are programmed only for interrupt event generation.                                                                                                                                         |
|         |            | 1<br>Reserved                                                                                                                                                                                                                                |
|         |            | 2<br>Interrupt and PPS Start/Stop<br>The Target Time registers are programmed for interrupt event and for starting or stopping the PPS output signal generation.                                                                             |
|         |            | 3<br>PPS Start/Stop Only<br>The Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.                                                                               |
| 4       | PPSEN      | Enable the flexible PPS output mode.<br>The EMAC_TM_PPCTL.PPSEN bit enables PPS operation. When set low, PPCTL field (bits [3:0]) controls frequency of Fixed PPS output. When set high, PPCTL field is used to command Flexible PPS output. |

**Table 23-149:** EMAC\_TM\_PPSTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:0     | PPSTL    | <p>PPS Frequency Control.</p> <p>When the EMAC_TM_PPSTL.PPS bit is disabled (=0, fixed PPS output), the EMAC_TM_PPSTL.PPS bits control the behavior of the PPS output signal. When the EMAC_TM_PPSTL.PPS bit is enabled (=1, flexible PPS output), the EMAC_TM_PPSTL.PPS bits function as PPSCMD. (See bit enumerations for PPS output frequency, rollover, and PPS commands.) Programming these bits with a non-zero value instructs the core to initiate an event. After the command is transferred or synchronized to the PTP clock domain, these bits gets cleared automatically. Software should ensure that these bits are programmed only when they are "all-zero". All values not shown in the bit enumerations are reserved.</p> <p>For more information about the EMAC_TM_PPSTL.PPS bits, see the register description.</p> |
|         |          | 0 CMD=No Command                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 CMD=START Single; BR=2kHz; DR=1kHz<br>This PPS command generates single pulse rising at start point defined in Target Time Registers and of duration defined in PPS Width Register. (Binary Rollover = 2 kHz; Digital Rollover = 1 kHz.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 2 CMD=START Pulse; BR=4kHz; DR=2kHz<br>This PPS command generates the train of pulses rising at the start point defined in the Target Time Registers and of duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands. (Binary Rollover = 4 kHz; Digital Rollover = 2 kHz.)                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 3 CMD=Cancel START; BR=8kHz; DR=4kHz<br>This PPS command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time. (Binary Rollover = 4 kHz; Digital Rollover = 4 kHz.)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Time Stamp Auxiliary TS Nano Seconds Register

The EMAC\_TM\_AUXSTMP\_NSEC register contains the low 32 bits (nano-seconds field) of the auxiliary time stamp.

EMAC\_TM\_AUXSTMP\_NSEC: Time Stamp Auxiliary TS Nano Seconds Register - R/W

Reset = 0x0000 0000

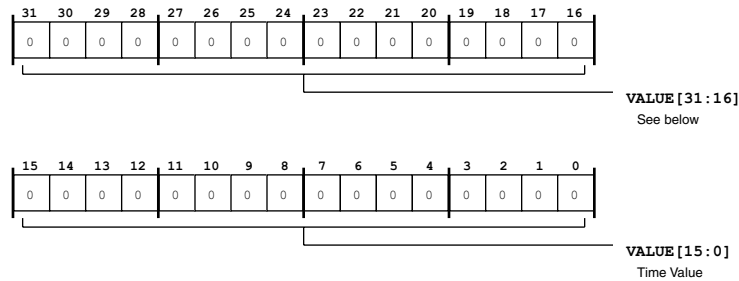


Figure 23-125: EMAC\_TM\_AUXSTMP\_NSEC Register Diagram

Table 23-150: EMAC\_TM\_AUXSTMP\_NSEC Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Time Value.             |

## Time Stamp Auxiliary TM Seconds Register

The EMAC\_TM\_AUXSTMP\_SEC register contains the low 32 bits of the seconds field of the auxiliary time stamp.

EMAC\_TM\_AUXSTMP\_SEC: Time Stamp Auxiliary TM Seconds Register - R/W

Reset = 0x0000 0000

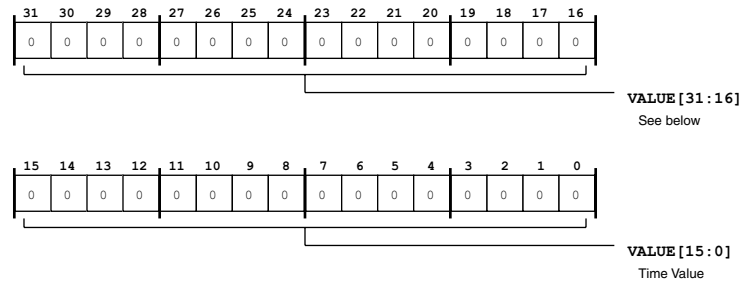


Figure 23-126: EMAC\_TM\_AUXSTMP\_SEC Register Diagram

**Table 23-151:** EMAC\_TM\_AUXSTMP\_SEC Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Time Value.             |

## Time Stamp PPS Interval Register

The EMAC\_TM\_PPSINTVL register contains the interval value for the time between rising edges (period) of PPS output.

**EMAC\_TM\_PPSINTVL: Time Stamp PPS Interval Register - R/W**

Reset = 0x0000 0000

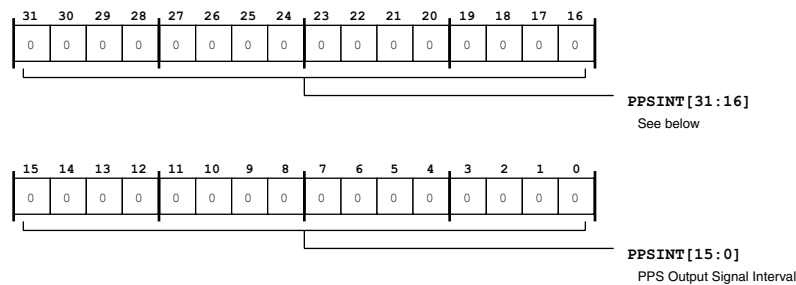


**Figure 23-127:** EMAC\_TM\_PPSINTVL Register Diagram

**Table 23-152:** EMAC\_TM\_PPSINTVL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | PPSINT   | PPS Output Signal Interval.<br>The EMAC_TM_PPSINTVL.PPSINT bits store the interval between the rising edges of PPS signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20ns), and desired interval between rising edges of PPS signal output is 100ns (that is, 5 units of sub-second increment value), then you should program value 4 (5-1) in this register. |

## PPS Width Register

The EMAC\_TM\_PPSWIDTH register contains the interval value for the time between a rising and the next falling edge (width) of PPS output.

EMAC\_TM\_PPSWIDTH: PPS Width Register - R/W

Reset = 0x0000 0000

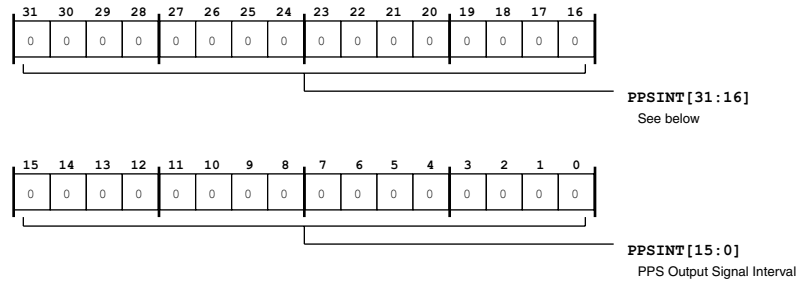


Figure 23-128: EMAC\_TM\_PPSWIDTH Register Diagram

Table 23-153: EMAC\_TM\_PPSWIDTH Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | PPSINT   | PPS Output Signal Interval.<br>The EMAC_TM_PPSWIDTH.PPSINT bits store the interval between the rising edges of PPS signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20ns), and desired interval between rising edges of PPS signal output is 100ns (that is, 5 units of sub-second increment value), then you should program value 4 (5-1) in this register. |

DMA Bus Mode Register

The EMAC\_DMA\_BUSMODE register selects the DMA bus operating modes for EMAC DMA.



**EMAC\_DMA\_BUSMODE: DMA Bus Mode Register - R/W**

Reset = 0x0002 0101

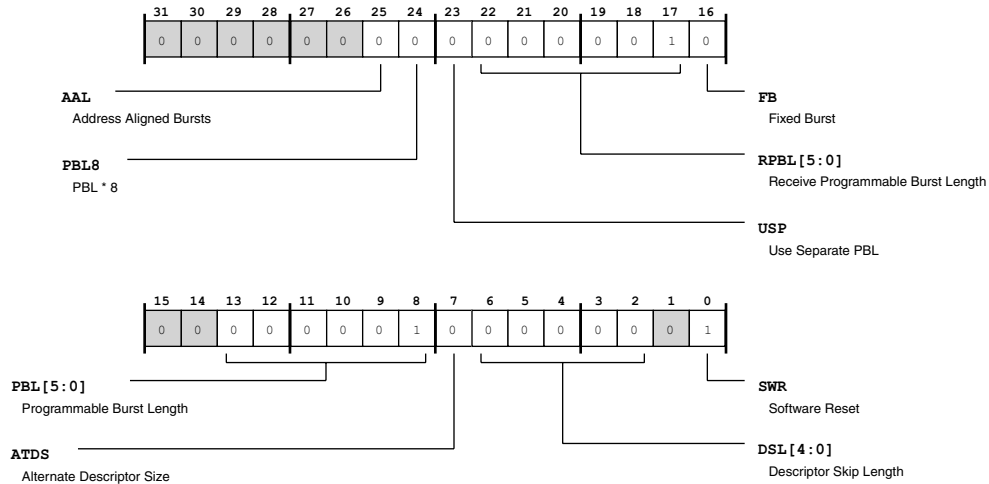


**Figure 23-129: EMAC\_DMA\_BUSMODE Register Diagram**

**Table 23-154: EMAC\_DMA\_BUSMODE Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | AAL      | Address Aligned Bursts.<br>The EMAC_DMA_BUSMODE.AAL bit, when set high and the FB bit equals 1, directs the SCB interface to generate all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffers start address) is not aligned, but subsequent bursts are aligned to the address. |
| 24      | PBL8     | PBL * 8.<br>The EMAC_DMA_BUSMODE.PBL8 bit, when set high, multiplies the PBL value programmed (bits [22:17] and bits [13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, and 32 beats depending on the PBL value.                                                                                                               |
| 23      | USP      | Use Separate PBL.<br>The EMAC_DMA_BUSMODE.USP bit, when set high, configures the RxDMA to use the value configured in bits [22:17] as PBL while the PBL value in bits [13:8] is applicable to TxDMA operations only.                                                                                                                             |

**Table 23-154:** EMAC\_DMA\_BUSMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22:17   | RPBL     | <p>Receive Programmable Burst Length.</p> <p>The EMAC_DMA_BUSMODE . RPBL bits indicate the maximum number of beats to be transferred in one RxDMA transaction. This is the maximum value that is used in a single block Read/Write. The RxDMA always attempts to burst as specified in RPBL every time it starts a Burst transfer on the host bus. RPBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. These bits are valid and applicable only when USP is set high.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 16      | FB       | <p>Fixed Burst.</p> <p>The EMAC_DMA_BUSMODE . FB bit controls whether the SCB Master interface performs fixed burst transfers or not. See bit 0 (UNDEF: SCB Undefined Burst Length) of the SCB Bus Mode register for detailed description.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 13:8    | PBL      | <p>Programmable Burst Length.</p> <p>The EMAC_DMA_BUSMODE . PBL bits indicate the maximum number of beats to be transferred in one DMA transaction. This will be the maximum value that is used in a single block Read/Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable for TxDMA transactions only.</p> <p>PBL-max limit = (FIFO size / 2) / 4.<br/> PBL-max limit (transmit) = 256 bytes / 2 / 4 = 32.<br/> PBL-max limit (receive) = 128 bytes / 2 / 4 = 16.</p> <p>Note that this PBL is at the DMA end. If PBL= 32 and if BLEN16 is enabled, the DMA automatically splits 32 bursts in to 2 x 16 bursts. If PBL = 8, and if BLEN16 is enabled, the max burst is limited to BLEN8. If PBL8 bit in DMA_BUSMODE register is set, the programmed PBL value is multiplied by 8 times internally. However, the result cannot be more than the above maximum limits specified above.</p> |

**Table 23-154:** EMAC\_DMA\_BUSMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | ATDS     | <p>Alternate Descriptor Size.</p> <p>The <code>EMAC_DMA_BUSMODE.ATDS</code> bit, when set, increases the size of the alternate descriptor to 32 bytes (8 DWORDS). This is required when the Advanced Time Stamp feature or Full IPC Offload Engine is enabled in the receiver. When reset, the descriptor size reverts back to 4 DWORDS (16 bytes). The enhanced descriptor is not required if the Advanced Time Stamp and IPC Full Checksum Offload features are not enabled. In such case, you can use the 16 bytes descriptor to save 4 bytes of memory.</p>                                                               |
| 6:2     | DSL      | <p>Descriptor Skip Length.</p> <p>The <code>EMAC_DMA_BUSMODE.DSL</code> bit specifies the number of 32-bit words to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When DSL value is equal to zero, then the descriptor table is taken as contiguous by the DMA, in Ring mode.</p>                                                                                                                                                                                                                                                   |
| 0       | SWR      | <p>Software Reset.</p> <p>The <code>EMAC_DMA_BUSMODE.SWR</code> bit, when set, directs the MAC DMA Controller to reset all MAC Subsystem internal registers and logic. It is cleared automatically after the reset operation has completed in all of the core clock domains. Read a 0 value in this bit before re-programming any register of the core. Note: The reset operation is completed only when all the resets in all the active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion.</p> |

## DMA Tx Poll Demand Register

The `EMAC_DMA_TXPOLL` register directs the EMAC to poll the transmit descriptor list.

EMAC\_DMA\_TXPOLL: DMA Tx Poll Demand Register - R/W

Reset = 0x0000 0000

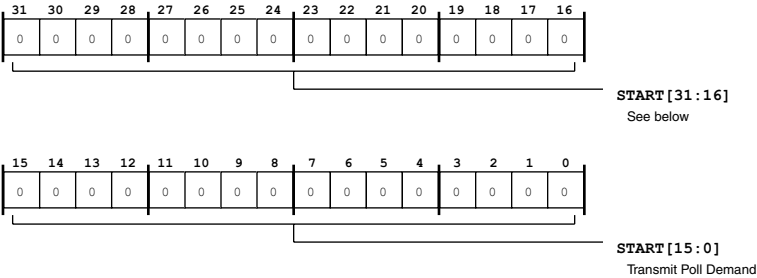


Figure 23-130: EMAC\_DMA\_TXPOLL Register Diagram

Table 23-155: EMAC\_DMA\_TXPOLL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | START    | Transmit Poll Demand.<br>The EMAC_DMA_TXPOLL . START bits, when written with any value, cause the DMA to read the current descriptor pointed to by Transmit Current Descriptor register. If that descriptor is not available (owned by application), transmission returns to the Suspend state and DMA Status register[2](Transmit buffer unavailable bit) is asserted. If the descriptor is available, transmission resumes. |

DMA Rx Poll Demand register

The EMAC\_DMA\_RXPOLL register directs the EMAC to poll the receive descriptor list.

EMAC\_DMA\_RXPOLL: DMA Rx Poll Demand register - R/W

Reset = 0x0000 0000

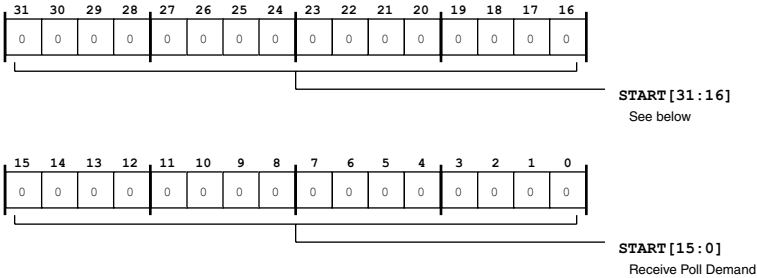


Figure 23-131: EMAC\_DMA\_RXPOLL Register Diagram

**Table 23-156:** EMAC\_DMA\_RXPOLL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | START    | Receive Poll Demand.<br>The EMAC_DMA_RXPOLL . START bits, when written with any value, cause the DMA to read the current descriptor pointed to by Receive Current Descriptor register. If that descriptor is not available (owned by application), reception returns to the Suspended state and DMA Status register[7](Receive buffer unavailable bit) is asserted. If the descriptor is available, the Receive DMA returns to the active state. |

## DMA Rx Descriptor List Address Register

The EMAC\_DMA\_RXDSC\_ADDR register holds the address for the DMA receive descriptor list. Writing to this Register is permitted only when reception is stopped. When stopped, this must be written to before the receive Start command is given. The processor can write to DMA\_RXDSC\_ADDR only when Rx DMA has stopped (bit 1, SR, =0 in Operation Mode Register). When stopped, it can be written with a new descriptor list address. When the processor sets the SR bit to 1, the DMA takes the newly programmed descriptor base address. If this register is not changed when the SR bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

### EMAC\_DMA\_RXDSC\_ADDR: DMA Rx Descriptor List Address Register - R/W

Reset = 0x0000 0000

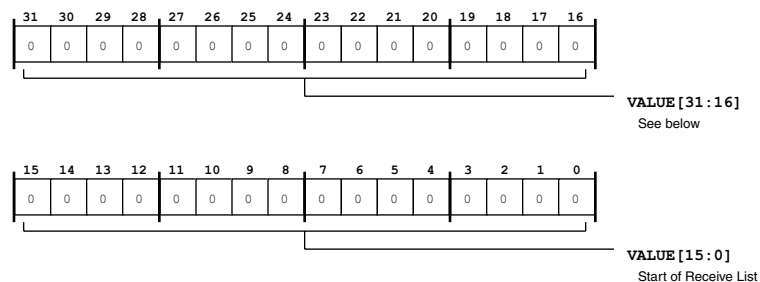


**Figure 23-132:** EMAC\_DMA\_RXDSC\_ADDR Register Diagram

**Table 23-157:** EMAC\_DMA\_RXDSC\_ADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Start of Receive List.<br>The EMAC_DMA_RXDSC_ADDR.VALUE bit field contains the base address of the First Descriptor in the Receive Descriptor list. The LSB bits [1:0] for the 32bit bus width are ignored and are taken as all-zero by the DMA internally. Therefore, these LSB bits are Read-Only (RO). |

## DMA Tx Descriptor List Address Register

The EMAC\_DMA\_TXDSC\_ADDR register holds the address for the DMA transmit descriptor list. The processor can write to this Register only when Tx DMA has stopped (bit 13, ST, =0 in Operation Mode Register). When stopped, this can be written with a new descriptor list address. When the processor sets the ST bit to 1, the DMA takes the newly programmed descriptor base address. If this Register is not changed when the ST bit is set to 0, then the DMA takes the descriptor address where it was stopped earlier.

**EMAC\_DMA\_TXDSC\_ADDR: DMA Tx Descriptor List Address Register - R/W**

Reset = 0x0000 0000

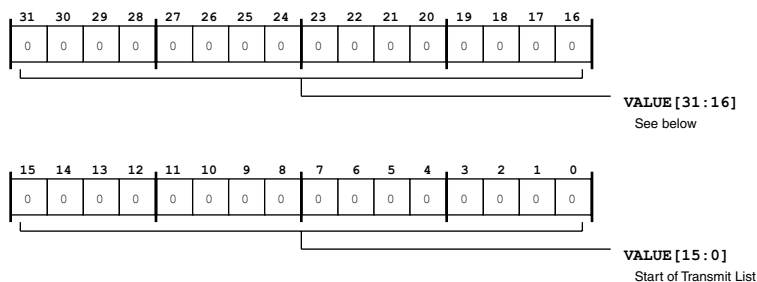


**Figure 23-133:** EMAC\_DMA\_TXDSC\_ADDR Register Diagram

**Table 23-158:** EMAC\_DMA\_TXDSC\_ADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Start of Transmit List.<br>The EMAC_DMA_TXDSC_ADDR.VALUE bit field contains the base address of the First Descriptor in the Transmit Descriptor list. The LSB bits [1:0] for 32bit bus width are ignored and are taken as all-zero by the DMA internally. Therefore, these LSB bits are Read-Only (RO). |

# DMA Status Register

The EMAC\_DMA\_STAT register indicates EMAC DMA status.

EMAC\_DMA\_STAT: DMA Status Register - R/W

Reset = 0x0000 0000

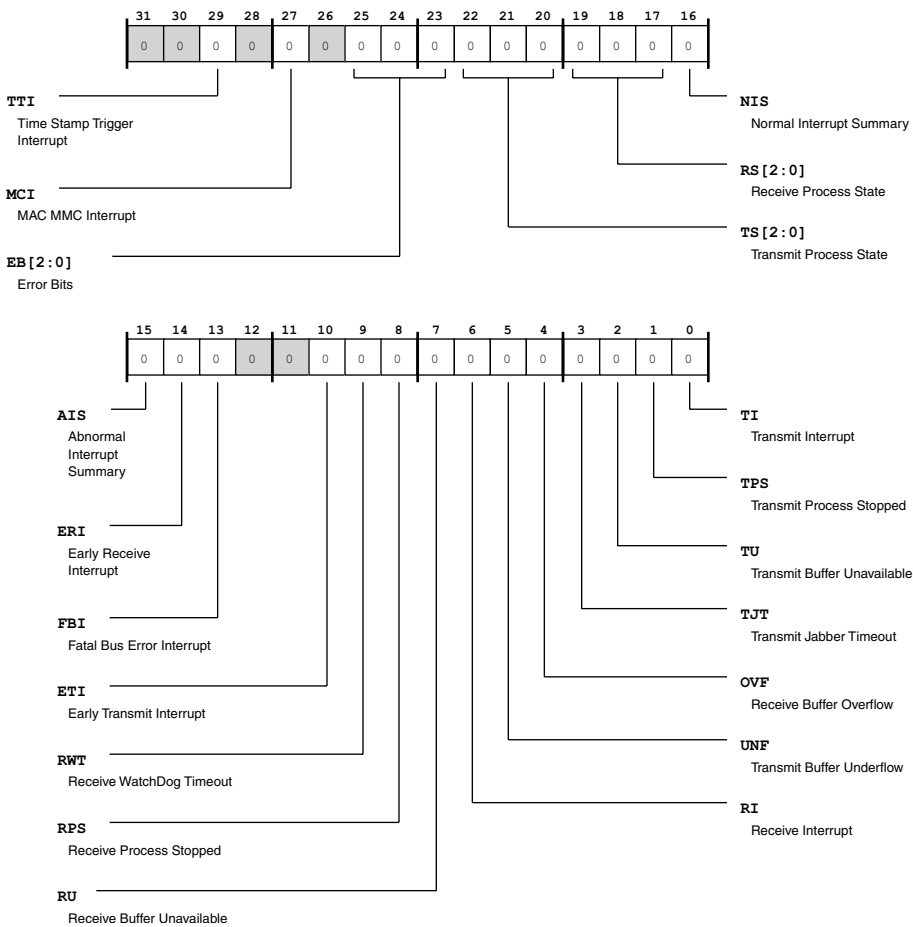


Figure 23-134: EMAC\_DMA\_STAT Register Diagram

Table 23-159: EMAC\_DMA\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29      | TTI      | Time Stamp Trigger Interrupt.<br>The EMAC_DMA_STAT.TTI bit indicates an interrupt event in the MAC core's Timestamp Generator block. The software must read the corresponding registers in the MAC core to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the MAC is high. |

**Table 23-159:** EMAC\_DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 27      | MCI      | MAC MMC Interrupt.<br>The <code>EMAC_DMA_STAT.MCI</code> bit reflects an interrupt event in the MMC module of the MAC core. The software must read the corresponding registers in the MAC core to get the exact cause of interrupt and clear the source of interrupt to make this bit as 1'b0. The interrupt signal from the MAC is high when this bit is high. |
| 25:23   | EB       | Error Bits.<br>The <code>EMAC_DMA_STAT.EB</code> bits indicate the type of error that caused a Bus Error (e.g., error response on the SCB interface). These bits are valid only when bit 13 (FBI: Fatal Bus Error Interrupt) is set. This field does not generate an interrupt.                                                                                 |
|         |          | 0Error during data buffer access, write transfer, RX DMA                                                                                                                                                                                                                                                                                                        |
|         |          | 1Error during data buffer access, write transfer, TX DMA                                                                                                                                                                                                                                                                                                        |
|         |          | 2Error during data buffer access, read transfer, RX DMA                                                                                                                                                                                                                                                                                                         |
|         |          | 3Error during data buffer access, read transfer, TX DMA                                                                                                                                                                                                                                                                                                         |
|         |          | 4Error during descriptor access, write transfer, RX DMA                                                                                                                                                                                                                                                                                                         |
|         |          | 5Error during descriptor access, write transfer, TX DMA                                                                                                                                                                                                                                                                                                         |
|         |          | 6Error during descriptor access, read transfer, RX DMA                                                                                                                                                                                                                                                                                                          |
|         |          | 7Error during descriptor access, read transfer, TX DMA                                                                                                                                                                                                                                                                                                          |



**Table 23-159:** EMAC\_DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------|
| 22:20   | TS       | Transmit Process State.<br>The EMAC_DMA_STAT.TS bits indicate the Transmit DMA state. This field does not generate an interrupt. |
|         |          | 0 Stopped; Reset or Stop Transmit Command issued                                                                                 |
|         |          | 1 Running; Fetching Transmit Transfer Descriptor                                                                                 |
|         |          | 2 Running; Waiting for status                                                                                                    |
|         |          | 3 Reading Data from host memory buffer and queuing it to TX buffer                                                               |
|         |          | 4 TIME_STAMP write state                                                                                                         |
|         |          | 5 Reserved                                                                                                                       |
|         |          | 6 Suspended; Transmit Descriptor Unavailable or TX Buffer Underflow                                                              |
|         |          | 7 Closing Transmit Descriptor                                                                                                    |

**Table 23-159:** EMAC\_DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 19:17   | RS       | Receive Process State.<br>The EMAC_DMA_STAT.RS bits indicate the Receive DMA state. This field does not generate an interrupt.                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0 Stopped: Reset or Stop Receive Command issued.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Running: Fetching Receive Transfer Descriptor.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 2 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 3 Running: Waiting for receive packet                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 4 Suspended: Receive Descriptor Unavailable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 5 Running: Closing Receive Descriptor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 6 TIME_STAMP write state                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 7 Running: Transferring RX packet data from RX buffer to host memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 16      | NIS      | Normal Interrupt Summary.<br>The value of the EMAC_DMA_STAT.NIS bit field is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Interrupt Enable Register: EMAC_DMA_STAT.TI, EMAC_DMA_STAT.TU, EMAC_DMA_STAT.RI, and EMAC_DMA_STAT.ERI. Only unmasked bits affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing a 1 to this bit) each time a corresponding bit that causes NIS to be set is cleared.                                                                |
| 15      | AIS      | Abnormal Interrupt Summary.<br>The value of the EMAC_DMA_STAT.AIS bit field is the logical OR of the following when the corresponding interrupt bits are enabled in DMA Interrupt Enable Register: EMAC_DMA_IEN.TPS, EMAC_DMA_IEN.TJT, EMAC_DMA_IEN.OVF, EMAC_DMA_IEN.UNF, EMAC_DMA_IEN.RU, EMAC_DMA_IEN.RPS, EMAC_DMA_IEN.RWT, EMAC_DMA_IEN.ETI, and EMAC_DMA_IEN.FBI. Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit that causes AIS to be set is cleared. |

**Table 23-159:** EMAC\_DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14      | ERI      | Early Receive Interrupt.<br>The EMAC_DMA_STAT.ERI bit indicates that the DMA had filled the first data buffer of the packet. RI bit of this register automatically clears this bit.                                                                                                                                                                                                                                                                                                                                                                            |
| 13      | FBI      | Fatal Bus Error Interrupt.<br>The EMAC_DMA_STAT.FBI bit indicates that a bus error occurred, as detailed in [25:23]. When this bit is set, the corresponding DMA engine disables all its bus accesses.                                                                                                                                                                                                                                                                                                                                                         |
| 10      | ETI      | Early Transmit Interrupt.<br>The EMAC_DMA_STAT.ETI bit indicates that the frame to be transmitted was fully transferred to the MFL Transmit FIFO.                                                                                                                                                                                                                                                                                                                                                                                                              |
| 9       | RWT      | Receive WatchDog Timeout.<br>The EMAC_DMA_STAT.RWT bit is asserted when a frame with a length greater than 2,048 bytes is received (10, 240 when Jumbo Frame mode is enabled).                                                                                                                                                                                                                                                                                                                                                                                 |
| 8       | RPS      | Receive Process Stopped.<br>The EMAC_DMA_STAT.RPS bit is asserted when the Receive Process enters the Stopped state.                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 7       | RU       | Receive Buffer Unavailable.<br>The EMAC_DMA_STAT.RU bit indicates that the Next Descriptor in the Receive List is owned by the application and cannot be acquired by the DMA. Receive Process is suspended. To resume processing Receive descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor was owned by the DMA. |
| 6       | RI       | Receive Interrupt.<br>The EMAC_DMA_STAT.RI bit indicates the completion of frame reception. Specific frame status information has been posted in the descriptor. Reception remains in the Running state.                                                                                                                                                                                                                                                                                                                                                       |
| 5       | UNF      | Transmit Buffer Underflow.<br>The EMAC_DMA_STAT.UNF bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.                                                                                                                                                                                                                                                                                                                                                       |

**Table 23-159:** EMAC\_DMA\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | OVF      | Receive Buffer Overflow.<br>The <code>EMAC_DMA_STAT.OVF</code> bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to application, the overflow status is set in <code>RDES0[11]</code> .                                                                                                                                                                                                |
| 3       | TJT      | Transmit Jabber Timeout.<br>The <code>EMAC_DMA_STAT.TJT</code> bit indicates that the Transmit Jabber Timer expired, meaning that the transmitter had been excessively active. The transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout <code>TDES0[14]</code> flag to assert.                                                                                                                     |
| 2       | TU       | Transmit Buffer Unavailable.<br>The <code>EMAC_DMA_STAT.TU</code> bit indicates that the Next Descriptor in the Transmit List is owned by the application and cannot be acquired by the DMA. Transmission is suspended. Bits[22:20] explain the Transmit Process state transitions. To resume processing transmit descriptors, the application should change the ownership of the bit of the descriptor and then issue a Transmit Poll Demand command. |
| 1       | TPS      | Transmit Process Stopped.<br>The <code>EMAC_DMA_STAT.TPS</code> bit is set when the transmission is stopped.                                                                                                                                                                                                                                                                                                                                           |
| 0       | TI       | Transmit Interrupt.<br>The <code>EMAC_DMA_STAT.TI</code> bit indicates that frame transmission is finished and <code>TDES1[31]</code> is set in the First Descriptor.                                                                                                                                                                                                                                                                                  |

## DMA Operation Mode Register

The `EMAC_DMA_OPMODE` register selects receive and transmit DMA operating modes.

**EMAC\_DMA\_OPMODE: DMA Operation Mode Register - R/W**

Reset = 0x0000 0000

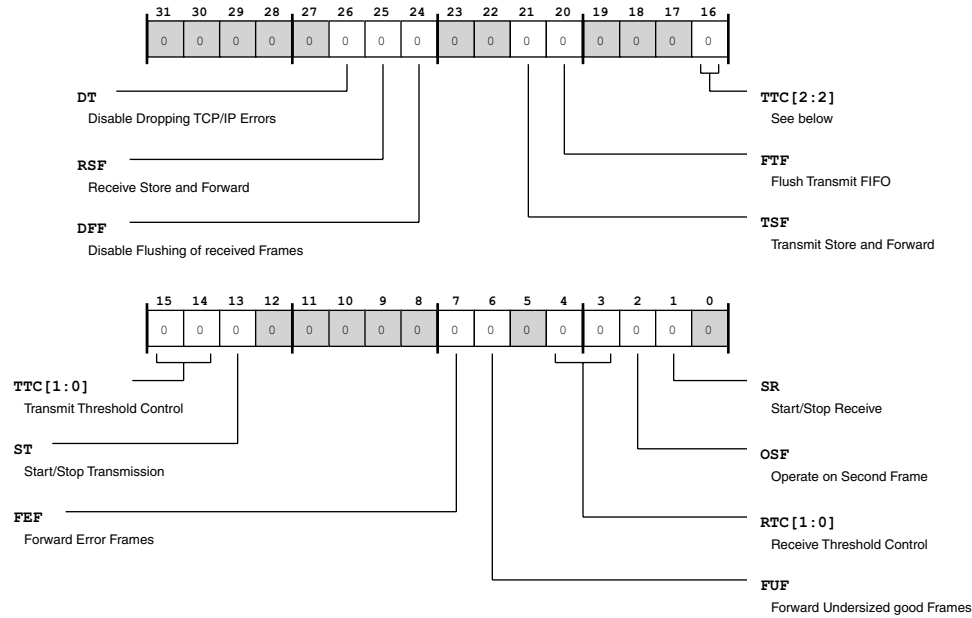


**Figure 23-135:** EMAC\_DMA\_OPMODE Register Diagram

**Table 23-160:** EMAC\_DMA\_OPMODE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | DT       | Disable Dropping TCP/IP Errors.<br>The EMAC_DMA_OPMODE.DT bit, when set, directs the core not to drop frames that only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors in the encapsulated payload only. When this bit is reset, all error frames are dropped if the FEF bit is reset. |
| 25      | RSF      | Receive Store and Forward.<br>The EMAC_DMA_OPMODE.RSF bit, when set, directs the MFL only to read a frame from the Rx FIFO after the complete frame has been written to it, ignoring RTC bits. When this bit is reset, the Rx FIFO operates in threshold mode, subject to the threshold specified by the RTC bits.                                                                                                     |

**Table 23-160:** EMAC\_DMA\_OPMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24      | DFF      | <p>Disable Flushing of received Frames.</p> <p>The EMAC_DMA_OPMODE.DFF bit, when set, directs the RxDMA not to flush any frames because of the unavailability of receive descriptors/buffers as it does normally when this bit is reset.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 21      | TSF      | <p>Transmit Store and Forward.</p> <p>The EMAC_DMA_OPMODE.TSF bit, when set, starts transmission when a full frame resides in the MFL Transmit FIFO. When this bit is set, the TTC values specified in Register 6[16:14] are ignored. This bit should be changed only when transmission is stopped.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 20      | FTF      | <p>Flush Transmit FIFO.</p> <p>The EMAC_DMA_OPMODE.FTF bit, when set, directs the transmit FIFO controller logic to reset to its default values and thus all data in the Tx FIFO is lost/flushed. This bit is cleared internally when the flushing operation is completed fully. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. Note: The flush operation completes only after emptying the TxFIFO of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. In order to complete this flush operation, the PHY transmit clock is required to be active.</p> |

**Table 23-160:** EMAC\_DMA\_OPMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16:14   | TTC      | <p>Transmit Threshold Control.</p> <p>The EMAC_DMA_OPMODE.TTC bits control the threshold level of the MFL Transmit FIFO. Transmission starts when the frame size within the MFL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when the TSF bit (Bit 21) is reset. "011" is not used.</p> |
|         |          | 0 64                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 128                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 2 192                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 256                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 4 40                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 5 32                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 6 24                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 16                                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 23-160:** EMAC\_DMA\_OPMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | ST       | <p>Start/Stop Transmission.</p> <p>The EMAC_DMA_OPMODE.ST bit, when set, places transmission in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Transmit Descriptor List Address, or from the position retained when transmission was stopped previously. If the current descriptor is not owned by the DMA, transmission enters the Suspended state and Transmit Buffer Unavailable (DMA Status register[2]) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Transmit Descriptor List Address register, then the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and becomes the current position when transmission is restarted. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.</p> |
| 7       | FEF      | <p>Forward Error Frames.</p> <p>The EMAC_DMA_OPMODE.FEF bit, when reset, directs the Rx FIFO to drop frames with error status (CRC error, collision error, giant frame, watchdog timeout, overflow). However, if the frames start byte (write) pointer is already transferred to the read controller side (in Threshold mode), then the frames are not dropped. When FEF bit is set, all frames except runt error frames are forwarded to the DMA. But when Rx FIFO overflows when a partial frame is written, then such frames are dropped even when FEF is set.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 6       | FUF      | <p>Forward Undersized good Frames.</p> <p>The EMAC_DMA_OPMODE.FUF bit, when set, directs the Rx FIFO to forward Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC). When reset, the Rx FIFO drops all frames of less than 64 bytes, unless it is already transferred because of lower value of Receive Threshold (e.g., RTC = 01).</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |



**Table 23-160:** EMAC\_DMA\_OPMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4:3     | RTC      | Receive Threshold Control.<br>The EMAC_DMA_OPMODE.RTC bits control the threshold level of the MFL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MFL Receive FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are transferred automatically. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. "11" is not used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 0 64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 32                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 2 96                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 3 128                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 2       | OSF      | Operate on Second Frame.<br>The EMAC_DMA_OPMODE.OSF bit, when set, instructs the DMA to process a second frame of Transmit data even before status for first frame is obtained.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 1       | SR       | Start/Stop Receive.<br>The EMAC_DMA_OPMODE.SR bit, when set, places the Receive process in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes incoming frames. Descriptor acquisition is attempted from the current position in the list, which is the address set by DMA Receive Descriptor List Address or the position retained when the Receive process was previously stopped. If no descriptor is owned by the DMA, reception is suspended and Receive Buffer Unavailable (DMA Status Register[7]) is set. The Start Receive command is effective only when reception has stopped. If the command was issued before setting DMA Receive Descriptor List Address Register, DMA behavior is unpredictable. When this bit is cleared, RxDMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state. |

## DMA Interrupt Enable Register

The EMAC\_DMA\_IEN register enables (unmasks) EMAC DMA interrupts.

**EMAC\_DMA\_IEN: DMA Interrupt Enable Register - R/W**

Reset = 0x0000 0000

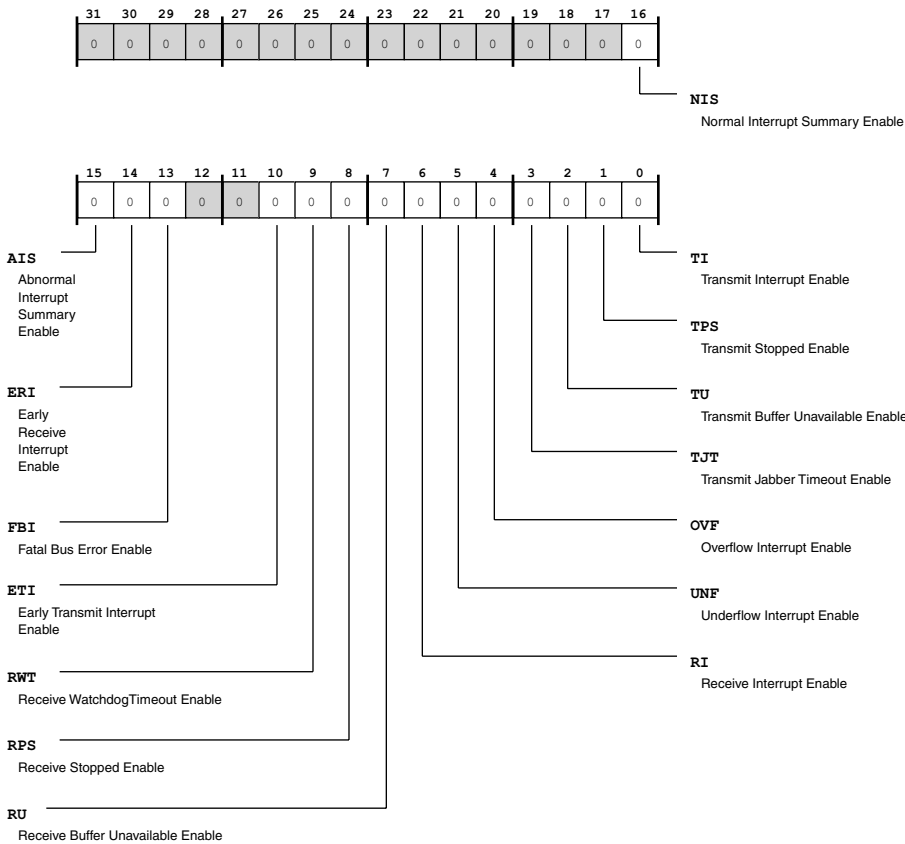


**Figure 23-136:** EMAC\_DMA\_IEN Register Diagram

**Table 23-161:** EMAC\_DMA\_IEN Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | NIS      | Normal Interrupt Summary Enable.<br>The EMAC_DMA_IEN.NIS bit, when set, enables a normal interrupt. When this bit is reset, a normal interrupt is disabled. This bit enables the following bits: EMAC_DMA_STAT.TI, EMAC_DMA_STAT.TU, EMAC_DMA_STAT.RI, and EMAC_DMA_STAT.ERI. |

**Table 23-161:** EMAC\_DMA\_IEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | AIS      | Abnormal Interrupt Summary Enable.<br>The EMAC_DMA_IEN.AIS bit, when set, enables an abnormal interrupt. When this bit is reset, an Abnormal Interrupt is disabled. This bit enables the following bits: EMAC_DMA_STAT.TPS, EMAC_DMA_STAT.TJT, EMAC_DMA_STAT.OVF, EMAC_DMA_STAT.RU, EMAC_DMA_STAT.RPS, EMAC_DMA_STAT.RWT, EMAC_DMA_STAT.ETI, and EMAC_DMA_STAT.FBI. |
| 14      | ERI      | Early Receive Interrupt Enable.<br>The EMAC_DMA_IEN.ERI bit, when set with Normal Interrupt Summary Enable in this register, enables the Early Receive Interrupt. When this bit is reset, Early Receive Interrupt is disabled.                                                                                                                                      |
| 13      | FBI      | Fatal Bus Error Enable.<br>The EMAC_DMA_IEN.FBI bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Fatal Bus Error Interrupt. When this bit is reset, Fatal Bus Error Enable Interrupt is disabled.                                                                                                                                 |
| 10      | ETI      | Early Transmit Interrupt Enable.<br>The EMAC_DMA_IEN.ETI bit, when this bit is set with an Abnormal Interrupt Summary Enable in this register, enables the Early Transmit Interrupt. When this bit is reset, Early Transmit Interrupt is disabled.                                                                                                                  |
| 9       | RWT      | Receive Watchdog Timeout Enable.<br>The EMAC_DMA_IEN.RWT bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Receive Watchdog Timeout Interrupt. When this bit is reset, Receive Watchdog Timeout Interrupt is disabled.                                                                                                             |
| 8       | RPS      | Receive Stopped Enable.<br>The EMAC_DMA_IEN.RPS bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Receive Stopped Interrupt is enabled. When this bit is reset, Receive Stopped Interrupt is disabled.                                                                                                                             |
| 7       | RU       | Receive Buffer Unavailable Enable.<br>The EMAC_DMA_IEN.RU bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Receive Buffer Unavailable Interrupt. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.                                                                                                    |

**Table 23-161:** EMAC\_DMA\_IEN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | RI       | Receive Interrupt Enable.<br>The EMAC_DMA_IEN.RI bit, when set with Normal Interrupt Summary Enable in this register, enables the Receive Interrupt. When this bit is reset, Receive Interrupt is disabled.                                                   |
| 5       | UNF      | Underflow Interrupt Enable.<br>The EMAC_DMA_IEN.UNF bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Transmit Underflow Interrupt. When this bit is reset, Underflow Interrupt is disabled.                                 |
| 4       | OVF      | Overflow Interrupt Enable.<br>The EMAC_DMA_IEN.OVF bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Receive Overflow Interrupt. When this bit is reset, Overflow Interrupt is disabled.                                     |
| 3       | TJT      | Transmit Jabber Timeout Enable.<br>The EMAC_DMA_IEN.TJT bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Transmit Jabber Timeout Interrupt. When this bit is reset, Transmit Jabber Timeout Interrupt is disabled.          |
| 2       | TU       | Transmit Buffer Unavailable Enable.<br>The EMAC_DMA_IEN.TU bit, when set with Normal Interrupt Summary Enable in this register, enables the Transmit Buffer Unavailable Interrupt. When this bit is reset, Transmit Buffer Unavailable Interrupt is disabled. |
| 1       | TPS      | Transmit Stopped Enable.<br>The EMAC_DMA_IEN.TPS bit, when set with Abnormal Interrupt Summary Enable in this register, enables the Transmission Stopped Interrupt. When this bit is reset, Transmission Stopped Interrupt is disabled.                       |
| 0       | TI       | Transmit Interrupt Enable.<br>The EMAC_DMA_IEN.TI bit, when set with Normal Interrupt Summary Enable in this register, enables the Transmit Interrupt. When this bit is reset, Transmit Interrupt is disabled.                                                |

## DMA Missed Frame Register

The EMAC\_DMA\_MISS\_FRM register contains counters for EMAC DMA missed frames and buffer overflows.

EMAC\_DMA\_MISS\_FRM: DMA Missed Frame Register - R/W

Reset = 0x0000 0000

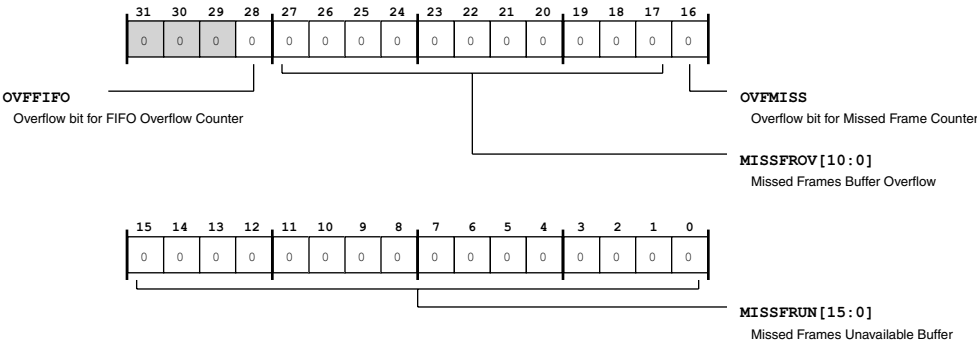


Figure 23-137: EMAC\_DMA\_MISS\_FRM Register Diagram

Table 23-162: EMAC\_DMA\_MISS\_FRM Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28      | OVFFIFO  | Overflow bit for FIFO Overflow Counter.<br>The EMAC_DMA_MISS_FRM.OVFFIFO bit holds the overflow bit for FIFO Overflow Counter.                                                               |
| 27:17   | MISSFROV | Missed Frames Buffer Overflow.<br>The EMAC_DMA_MISS_FRM.MISSFROV bits indicate the number of frames missed by the application due to buffer overflow.                                        |
| 16      | OVFMIS   | Overflow bit for Missed Frame Counter.<br>The EMAC_DMA_MISS_FRM.OVFMIS bit holds the overflow bit for the Missed Frame Counter.                                                              |
| 15:0    | MISSFRUN | Missed Frames Unavailable Buffer.<br>The EMAC_DMA_MISS_FRM.MISSFRUN bits indicate the number of frames missed by the controller because of the Application Receive Buffer being unavailable. |

DMA Rx Interrupt Watch Dog Register

The EMAC\_DMA\_RXIWDOG register contains the timeout value for the EMAC DMA receive interrupt watch dog timer.

EMAC\_DMA\_RXIWDOG: DMA Rx Interrupt Watch Dog Register - R/W

Reset = 0x0000 0000

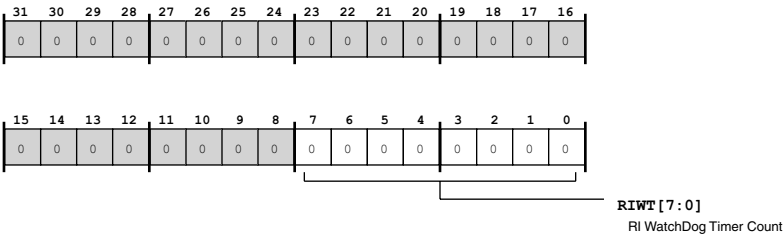


Figure 23-138: EMAC\_DMA\_RXIWDOG Register Diagram

Table 23-163: EMAC\_DMA\_RXIWDOG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | RIWT     | RI WatchDog Timer Count.<br>The EMAC_DMA_RXIWDOG.RIWT bit field indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the RxDMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watch-dog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame. |

DMA SCB Bus Mode Register

The EMAC\_DMA\_BMMODE register selects EMAC DMA system cross bar bus mode features.

**EMAC\_DMA\_BMMODE: DMA SCB Bus Mode Register - R/W**

Reset = 0x0011 0001

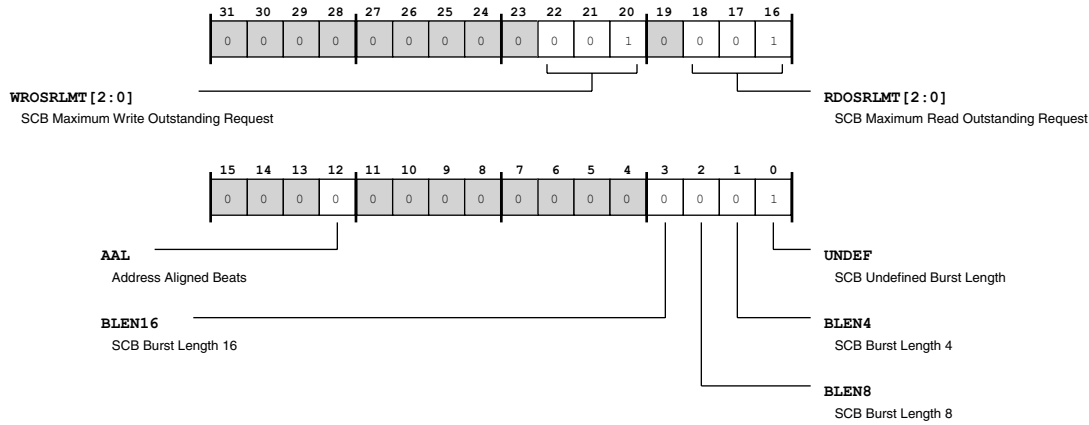


**Figure 23-139: EMAC\_DMA\_BMMODE Register Diagram**

**Table 23-164: EMAC\_DMA\_BMMODE Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22:20   | WROSRLMT | SCB Maximum Write Outstanding Request.<br>The EMAC_DMA_BMMODE.WROSRLMT bit field's value limits the maximum outstanding request on the SCB write interface. Maximum outstanding requests = WR_OSR_LMT+1. EMAC-SCB supports up to 4 outstanding write requests. |
| 18:16   | RDOSRLMT | SCB Maximum Read Outstanding Request.<br>The EMAC_DMA_BMMODE.RDOSRLMT bit field's value limits the maximum outstanding request on the SCB read interface. Maximum outstanding requests = RD_OSR_LMT+1. EMAC-SCB supports up to 4 outstanding read requests.    |
| 12      | AAL      | Address Aligned Beats.<br>The EMAC_DMA_BMMODE.AAL bit (read-only) reflects the AAL bit of DMA Bus mode register. When this bit is set to 1, EMAC-SCB performs address-aligned burst transfers on both read and write channels.                                 |
| 3       | BLEN16   | SCB Burst Length 16.<br>The EMAC_DMA_BMMODE.BLEN16 bit, when set (or when UNDEF is set to 1), directs the EMAC-SCB to select a burst length of 16 on the SCB master interface.                                                                                 |

Table 23-164: EMAC\_DMA\_BMMODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | BLN8     | SCB Burst Length 8.<br>The EMAC_DMA_BMMODE.BLN8 bit, when set (or when UNDEF is set to 1), directs the EMAC-SCB to select a burst length of 8 on the SCB master interface.                                                                                                                                                                                                                                                                                |
| 1       | BLN4     | SCB Burst Length 4.<br>The EMAC_DMA_BMMODE.BLN4 bit, when set (or when UNDEF is set to 1), directs the EMAC-SCB to select a burst length of 4 on the SCB master interface.                                                                                                                                                                                                                                                                                |
| 0       | UNDEF    | SCB Undefined Burst Length.<br>The EMAC_DMA_BMMODE.UNDEF bit (read-only) indicates the complement (invert) value of FB bit in DMA Bus Mode Register. When this bit is set to 1, the EMAC-SCB is allowed to perform any burst length equal to or below the maximum allowed burst length as programmed in bits[3:1]. When this bit is set to 0, the EMAC-SCB is allowed to perform only fixed burst lengths as indicated by 16/8/4, or a burst length of 1. |

DMA SCB Status Register

The EMAC\_DMA\_BMSTAT register indicates EMAC DMA system cross bar status.

EMAC\_DMA\_BMSTAT: DMA SCB Status Register - R/W

Reset = 0x0000 0000

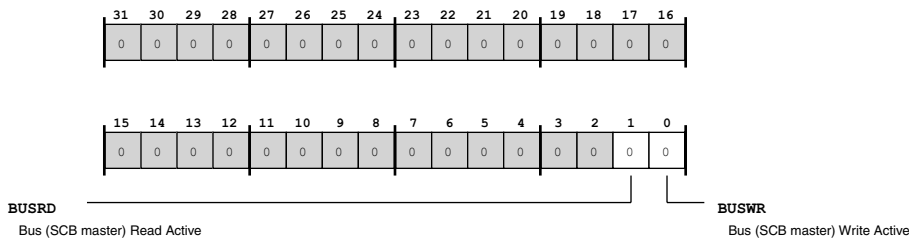


Figure 23-140: EMAC\_DMA\_BMSTAT Register Diagram



**Table 23-165:** EMAC\_DMA\_BMSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | BUSRD    | Bus (SCB master) Read Active.<br>The EMAC_DMA_BMSTAT.BUSRD bit, when high, indicates that SCB Master's read channel is active and transferring data.   |
| 0       | BUSWR    | Bus (SCB master) Write Active.<br>The EMAC_DMA_BMSTAT.BUSWR bit, when high, indicates that SCB Master's write channel is active and transferring data. |

## DMA Tx Descriptor Current Register

The EMAC\_DMA\_TXDSC\_CUR register contains the current DMA transmit descriptor.

**EMAC\_DMA\_TXDSC\_CUR: DMA Tx Descriptor Current Register - R/W**

Reset = 0x0000 0000

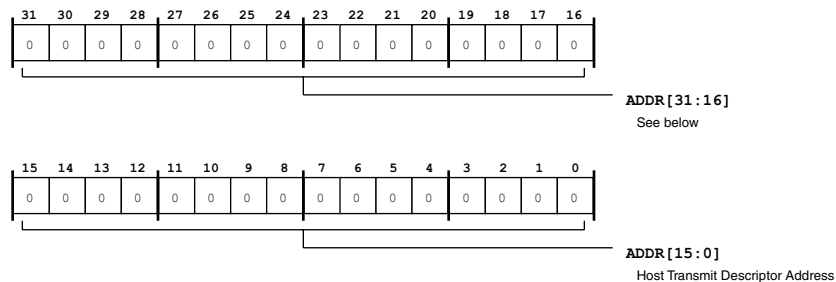


**Figure 23-141:** EMAC\_DMA\_TXDSC\_CUR Register Diagram

**Table 23-166:** EMAC\_DMA\_TXDSC\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                               |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | ADDR     | Host Transmit Descriptor Address.<br>The EMAC_DMA_TXDSC_CUR.ADDR bit field points to the start address of the current Transmit Descriptor read by the DMA. Pointer updated by DMA during operation. Cleared on Reset. |

## DMA Rx Descriptor Current Register

The EMAC\_DMA\_RXDSC\_CUR register contains the current DMA receive descriptor.

EMAC\_DMA\_RXDSC\_CUR: DMA Rx Descriptor Current Register - R/W

Reset = 0x0000 0000

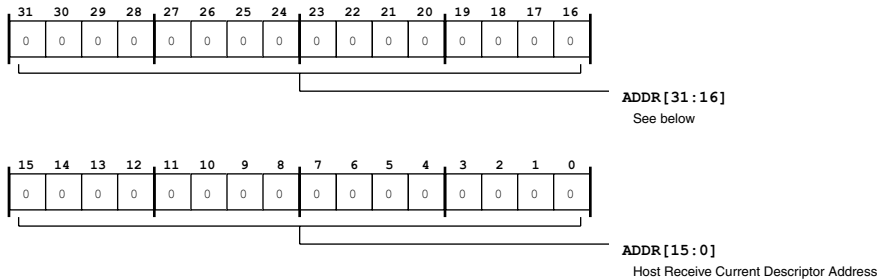


Figure 23-142: EMAC\_DMA\_RXDSC\_CUR Register Diagram

Table 23-167: EMAC\_DMA\_RXDSC\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | ADDR     | Host Receive Current Descriptor Address.<br>The EMAC_DMA_RXDSC_CUR.ADDR bit field points to the start address of the current Receive Descriptor read by the DMA. Pointer updated by DMA during operation. Cleared on Reset. |

DMA Tx Buffer Current Register

The EMAC\_DMA\_TXBUF\_CUR register holds the pointer to the current transmit DMA buffer.

EMAC\_DMA\_TXBUF\_CUR: DMA Tx Buffer Current Register - R/W

Reset = 0x0000 0000

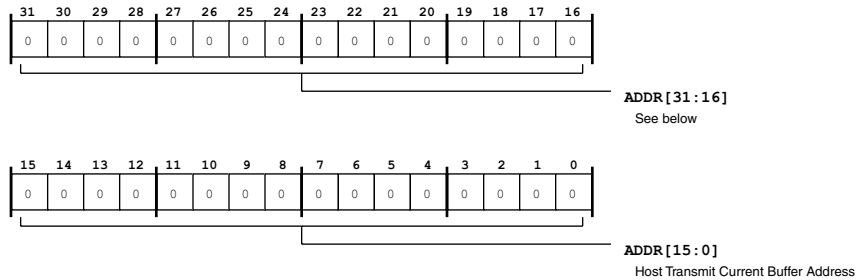


Figure 23-143: EMAC\_DMA\_TXBUF\_CUR Register Diagram

Table 23-168: EMAC\_DMA\_TXBUF\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | ADDR     | Host Transmit Current Buffer Address.<br>The EMAC_DMA_TXBUF_CUR.ADDR bit field points to the current Transmit Buffer Address being read by the DMA. Pointer updated by DMA during operation. Cleared on Reset. |

DMA Rx Buffer Current Register

The EMAC\_DMA\_RXBUF\_CUR register holds the pointer to the current receive DMA buffer.

EMAC\_DMA\_RXBUF\_CUR: DMA Rx Buffer Current Register - R/W

Reset = 0x0000 0000

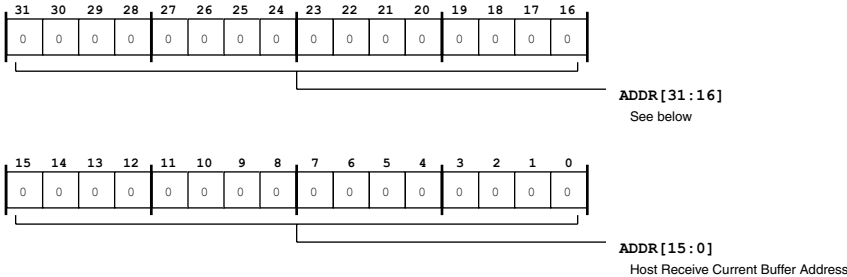


Figure 23-144: EMAC\_DMA\_RXBUF\_CUR Register Diagram

Table 23-169: EMAC\_DMA\_RXBUF\_CUR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | ADDR     | Host Receive Current Buffer Address.<br>The EMAC_DMA_RXBUF_CUR.ADDR bit field points to the current Receive Buffer address being read by the DMA. Pointer updated by DMA during operation. Cleared on Reset. |



## 24 Removable Storage Interface (RSI)

The removable storage interface (RSI) controller is a fast, synchronous peripheral that uses various protocols to communicate with MMC, SD, and SDIO cards as well as CE-ATA hard drives. The RSI is compatible with the following protocols.

- MMC (Multimedia Card) bus protocol
- SD (Secure Digital) bus protocol
- SDIO (Secure Digital Input Output) bus protocol
- CE-ATA (Consumer Electronic ATA)

All of these storage solutions use similar interface protocols. The main difference between MMC and SD support is the initialization sequence. The main difference between SD and SDIO support is the use of interrupt and read wait signals for SDIO. CE-ATA devices require handling of larger block sizes of 4K bytes and implement a device interrupt scheme known as the command completion signal (CCS).

**NOTE:** The RSI does not support the SPI bus protocol

### RSI Features

RSI includes the following features:

- Support for a single SD or SDIO card
- Support for a single MMC device (removable or embedded)
- Support for 1- and 4-bit SD modes (SPI mode is not supported)
- Support for 1-, 4-, and 8-bit MMC modes (SPI mode is not supported)
- Support for 4- and 8-bit CE-ATA devices
- Programmable clock frequency generated from SCLK
- Card detection capabilities
- SDIO interrupt and read wait features
- Command completion signal recognition and disable for CE-ATA device support
- High-capacity card support such as SDHC implemented within software
- 512-bit transmit/receive FIFO

- DMA channel with 32-bit DMA access bus

**Table 24-1:** RSI Specifications

| Feature                  | Availability |
|--------------------------|--------------|
| <b>Protocol</b>          |              |
| Master-Capable           | Yes          |
| Slave-Capable            | No           |
| Transmission Simplex     | Yes          |
| Transmission Half-Duplex | Yes          |
| Transmission Full-Duplex | No           |
| <b>Access Type</b>       |              |
| Data Buffer              | Yes          |
| Core Data Access         | Yes          |
| DMA Data Access          | Yes          |
| DMA Channels             | 1            |
| DMA Descriptor           | Yes          |
| Boot Capable             | Yes          |
| Local Memory             | No           |
| Clock Operation          | SCLK0/2      |

## RSI Functional Description

The following sections provide details on the RSI functionality.

## RSI Register List

**Table 24-2:** RSI Module Registers

| Name              | Description                               |
|-------------------|-------------------------------------------|
| RSI_CTL           | Control Register                          |
| RSI_ARG           | Argument Register                         |
| RSI_CMD           | Command Register                          |
| RSI_RESP_CMD      | Response Command Register                 |
| RSI_RESP0         | Response 0 Register                       |
| RSI_RESP1         | Response 1 Register                       |
| RSI_RESP2         | Response 2 Register                       |
| RSI_RESP3         | Response 3 Register                       |
| RSI_DATA_TMR      | Data Timer Register                       |
| RSI_DATA_LEN      | Data Length Register                      |
| RSI_DATA_CTL      | Data Control Register                     |
| RSI_DATA_CNT      | Data Count Register                       |
| RSI_XFRSTAT       | Status Register                           |
| RSI_XFRSTAT_CLR   | Status Clear Register                     |
| RSI_XFR_IMSK0     | Interrupt 0 Mask Register                 |
| RSI_XFR_IMSK1     | Interrupt 1 Mask Register                 |
| RSI_FIFO_CNT      | FIFO Counter Register                     |
| RSI_CEATA         | This register contains bit to dis CCS gen |
| RSI_BOOT_TCNTR    | Boot Timing Counter Register              |
| RSI_BACK_TOUT     | Boot Acknowledge Timeout Register         |
| RSI_SLP_WKUP_TOUT | Sleep Wakeup Timeout Register             |
| RSI_BLKSZ         | Block Size Register                       |

**Table 24-2:** RSI Module Registers (Continued)

| Name        | Description                        |
|-------------|------------------------------------|
| RSI_FIFO    | Data FIFO Register                 |
| RSI_STAT0   | Exception Status Register          |
| RSI_IMSK0   | Exception Mask Register            |
| RSI_CFG     | Configuration Register             |
| RSI_RD_WAIT | Read Wait Enable Register          |
| RSI_PID0    | Peripheral Identification Register |
| RSI_PID1    | Peripheral Identification Register |
| RSI_PID2    | Peripheral Identification Register |
| RSI_PID3    | Peripheral Identification Register |

## ADSP-BF60x RSI Interrupt List

**Table 24-3:** ADSP-BF60x RSI Interrupt List

| Description      | ID | DMA Channel | Sensitivity |
|------------------|----|-------------|-------------|
| RSI0 DMA Channel | 61 | 10          | LEVEL       |
| RSI0 Interrupt 0 | 62 |             | LEVEL       |
| RSI0 Interrupt 1 | 63 |             | LEVEL       |

## ADSP-BF60x RSI Trigger List

**Table 24-4:** Masters ADSP-BF60x RSI Trigger List

| Description      | ID | Sensitivity |
|------------------|----|-------------|
| RSI0 DMA Channel | 30 | PULSE/EDGE  |
|                  |    |             |



**Table 24-5:** Slaves ADSP-BF60x RSI Trigger List

| Description      | ID | Sensitivity |
|------------------|----|-------------|
| RSI0 DMA Channel | 30 |             |
|                  |    |             |

## RSI Clock Configuration

The RSI is a fast, synchronous peripheral with a programmable clock frequency that is supplied by the *RSI\_CLK* signal. The interface between the RSI and the internal buses operates at *SCLKx* frequency. Communication between the clock domain that is supplied externally from the RSI on the *RSI\_CLK* signal and the RSI access to the internal busses is accomplished using synchronizers in the RSI module. The *RSI\_CLK* frequency is configured by the 8-bit *CLKDIV* field and the *BYPASS* bit of the *RSI\_CTL* register.

If *BYPASS* is set, the clock frequency driven on the *RSI\_CLK* signal is derived directly from *SCLKx*. If *BYPASS* is cleared, the clock divider logic provides an *RSI\_CLK* frequency as shown below, where *CLKDIV* is an 8-bit value ranging between 0 and 255

$$RSI\_CLK = SCLKx / 2^{(CLKDIV+1)}$$

The *RSI\_CLK* output is enabled or disabled by the *CLKEN* bit in the *RSI\_CTL* register and a power save feature is implemented by setting *PWRSAVE*, which disables the *RSI\_CLK* output when there are no transfers taking place on the RSI interface.

## RSI Interface Configuration

The RSI supports multiple card types under various protocols. Different card types may require slightly different interface configurations.

The command signal on MMC cards operates in two different modes depending upon the operating mode of the card. During the card identification mode, the command signal operates in open-drain configuration; but when the card enters data transfer mode, the signal is configured to push-pull mode.

**NOTE:** The internal pull-up resistor of the *RSI\_CMD* signal is only intended to keep the signal from floating. The internal pull-up resistor is not sufficient during the card identification phase when the MMC card *RSI\_CMD* signal is operating in open-drain mode. If support for MMC devices is required, an external pull-up resistor should be added to the *RSI\_CMD* signal as detailed in the JEDEC standard.

The bus width used for the data transfers is configurable to 1-bit, 4-bits, or 8-bits using the `BUSWID` field in the `RSI_CTL` register.

To stop signals from floating when no card is inserted or during times when all card drivers are in a high-impedance mode, various pull-up and pull-down resistor configurations can be enabled on the `RSI_DATAx` signals. The `RSI_CFG` register provides the following options.

- Enable or disable a pull-up resistor on the `RSI_DATA3` signal
- Enable or disable pull-up resistors on the `RSI_DATA7` through `RSI_DATA4` and `RSI_DATA2` through `RSI_DATA0` signals.

## Card Detection

The RSI allows for software to detect when a card is inserted into its slot. There are a number of ways that this card detection can be performed.

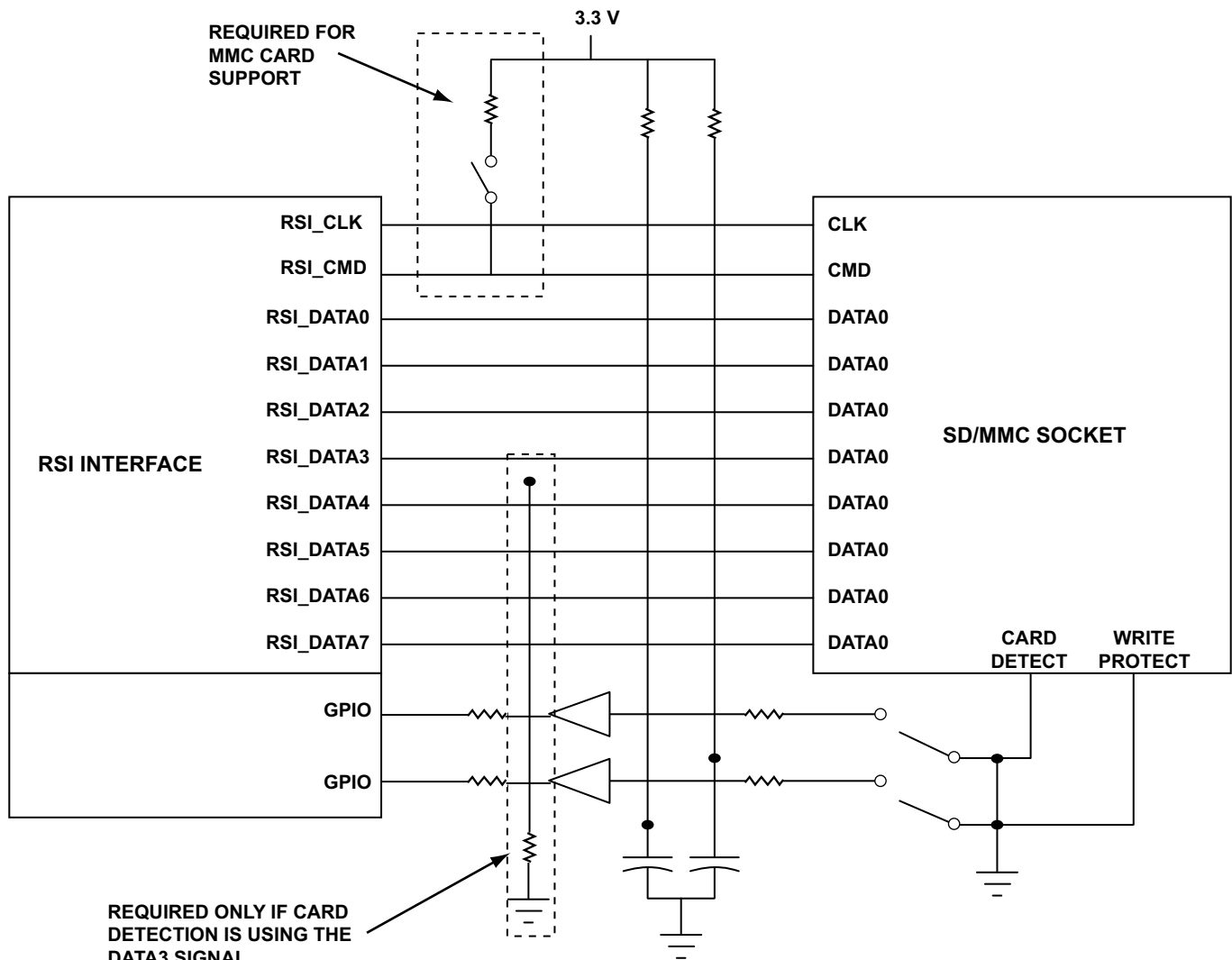


**Figure 24-1:** RSI Socket Interface

1. Using RSI\_DATA3 – SD and SDIO cards use an internal pull-up resistor on the RSI\_DATA3 line as a card detect signal to indicate to the host that a card is present. In order to use the RSI\_DATA3 signal for card detection, an external pull down resistor should be added to the pin to pull the pin low during the time a card is not inserted. When a card is inserted into the slot, a rising edge is detected on RSI\_DATA3 by the RSI and **SDCARD** bit in the **RSI\_STAT0** register is set. Once the card has been correctly identified, the **SDCARD** interrupt should be cleared and disabled. The pull-up resistor in the SD card should be disabled by issuing the **SET\_CLR\_CARD\_DETECT** command. When using the RSI\_DATA3 signal for card detection with an external pull-down resistor, do not enable the internal pull-up resistor by setting **DAT3PUP** in **RSI\_CFG** register.
2. Using SD/MMC Socket and GPIO Interrupt - The recommended method of detecting the insertion of a card is to use the card detect feature that is made available through most sockets. Sockets supporting this feature can have the card detect pin de-bounced and connected to a GPIO pin in order to allow not only interrupt driven card detection but also interrupt driven card removal. This is the most reliable

and efficient method of detecting the insertion and removal of a card as some MMC devices may not implement the card detect pull-up resistor on the *RSI\_DATA3* signal. Once a card is detected, the GPIO pin can have the interrupt level inverted to then generate an interrupt on card removal.

3. Using Software Polling – The final approach to detecting the insertion and removal of a card is to simply use software polling. Software would poll the slot periodically using the card identification commands for the supported card types. Once a card is inserted, this will result in valid responses being sent back to the host; when the card is removed, command and data timeout errors will be observed.

## RSI Power Saving Configuration

The RSI requires two internal clock signals that are derived directly from SCLKx. One of these clock signals is routed to the clock divider and generates the RSI\_CLK clock.

These clocks must be enabled by setting *CLKSEN* in the *RSI\_CFG* register for the RSI to function. Clearing *CLKSEN* disables the RSI regardless of other RSI clock configurations. The RSI\_CLK signal can be enabled or disabled using *CLKEN* in the *RSI\_CTL* register.

Additional power saving options can be implemented by setting *PWRSAVE* in the *RSI\_CTL* register which disables the RSI\_CLK output when there are no transfers taking place on the RSI interface. These configurations are shown in the table "RSI Power Saving Configurations".

**Table 24-6:** RSI Power Saving Configurations

| CLKSEN(RSI_CFG) | CLKEN(RSI_CTL) | PWRSAVE(RSI_CTL) | RSI State | RSI_CLK Output                                  |
|-----------------|----------------|------------------|-----------|-------------------------------------------------|
| 0               | 0              | 0                | Disabled  | No clock                                        |
| 0               | 0              | 1                | Disabled  | No clock                                        |
| 0               | 1              | 0                | Disabled  | No clock                                        |
| 0               | 1              | 1                | Disabled  | No clock                                        |
| 1               | 0              | 0                | Enabled   | No clock                                        |
| 1               | 0              | 1                | Enabled   | No clock                                        |
| 1               | 1              | 0                | Enabled   | Continuous clock <sup>1</sup>                   |
| 1               | 1              | 1                | Enabled   | Clock only driven during transfers <sup>1</sup> |

<sup>1</sup> The *PWRON* field of the *RSI\_CFG* register must be set to 0x3. If *PWRON* is cleared, the clock will not output.

## RSI Command-Response Interface

The RSI sends commands to and receives responses from the card using the RSI\_CMD signal. The command to be sent to the card is issued by writing to the RSI\_CMD register. This register contains a 6-bit `IDX` field that contains the command index to be sent to the card. The command index provides support for a total of 64 commands—0 (CMD0) to 63 (CMD63). Some commands require an argument to be sent along with the command, such as an address for a read transaction. An argument is always sent with the command and it is the responsibility of the card to either ignore or use the argument field based on the command that is received. The argument sent with the command is provided using the RSI\_ARG register.

All command transfers are protected by a 7-bit cyclic redundancy check (CRC) code, more commonly referred to as a CRC7 checksum. This allows for transmission errors to be detected and for the command to be re-issued to the card in the event of an error. All commands sent to the card are composed of 48-bits as shown in the table below.

**Table 24-7:** RSI Command Format

| Bit Position | Width | Value | Description     |
|--------------|-------|-------|-----------------|
| 47           | 1     | 0     | Start bit       |
| 46           | 1     | 1     | Transmitter bit |
| [45:40]      | 6     | -     | Command index   |
| [39:8]       | 32    | -     | Argument        |
| [7:1]        | 7     | -     | CRC7 checksum   |
| 0            | 1     | 1     | End bit         |

The RSI\_CMD register also provides configuration information about whether a response is to be expected back from the card and the type of response.

The RSI can be configured using the `RSP` and `LRSP` fields of the RSI\_CMD register to expect the following response types.

- No response
- Short response (see the table "RSI Short Response Format")
- Long response (see the table "RSI Long Response Format")

**Table 24-8:** RSI Short Response Format

| Bit Position | Width | Value | Description |
|--------------|-------|-------|-------------|
| 47           | 1     | 0     | Start bit   |

**Table 24-8:** RSI Short Response Format (Continued)

| Bit Position | Width | Value | Description                                      |
|--------------|-------|-------|--------------------------------------------------|
| 46           | 1     | 0     | Transmitter bit                                  |
| [45:40]      | 6     | -     | Command index or check bits <sup>1</sup>         |
| [39:8]       | 32    | -     | Card status, register contents or argument field |
| [7:1]        | 7     | -     | CRC7 checksum or check bits <sup>2</sup>         |
| 0            | 1     | 1     | End bit                                          |

1 Responses that do not contain the command index have b#111111 in the check bits field.

2 Responses that do not contain a CRC7 check sum have b#111111 in the check bits field.

**Table 24-9:** RSI Long Response Format

| Bit Position | Width | Value  | Description                                            |
|--------------|-------|--------|--------------------------------------------------------|
| 135          | 1     | 0      | Start bit                                              |
| 134          | 1     | 0      | Transmitter bit                                        |
| 133:128      | 6     | 111111 | Check bits <sup>1</sup>                                |
| 127:1        | 127   | -      | Register contents including internal CRC7 <sup>2</sup> |
| 0            | 1     | 1      | End bit                                                |

1 Responses that do not contain the command index have b#111111 in the check bits field.

2 Responses that do not contain a CRC7 check sum have b#111111 in the check bits field.

Like the commands, all responses are sent on the RSI\_CMD signal. A response always has a 0 start bit followed by a 0 transmission bit to indicate the transfer is from card to the RSI. Unlike the commands issued by the RSI, not all responses are protected by a CRC7 checksum. Refer to the appropriate specification for full details on the response formats for a specific device and whether they are protected by a CRC7 checksum.

When a short response is received, the response is broken down by the RSI. The 32-bit field containing bits[39:8] is stored to RSI\_RESP0, where bit 39 of the response corresponds to bit 31 of RSI\_RESP0 and bit 8 of the response to bit 0 of RSI\_RESP0. Bits[45:40] of the response are stored to the RESP\_CMD field of the RSI\_RESP\_CMD register.

For a long response, bits [127:1] of the response are stored in RSI\_RESP3–0. Bit 31 of RSI\_RESP0 contains the most significant bit (bit 127) of the response and bit 0 of RSI\_RESP3 contains bit 1 of the response. Bit 31 of RSI\_RESP3 is always zero.

The following figure shows the command path state machine. For the state machine to be active, the RSI must be enabled through CLKEN bit of the RSI\_CTL register. Disabling the clocks to the RSI results in the state machine returning to the IDLE state.

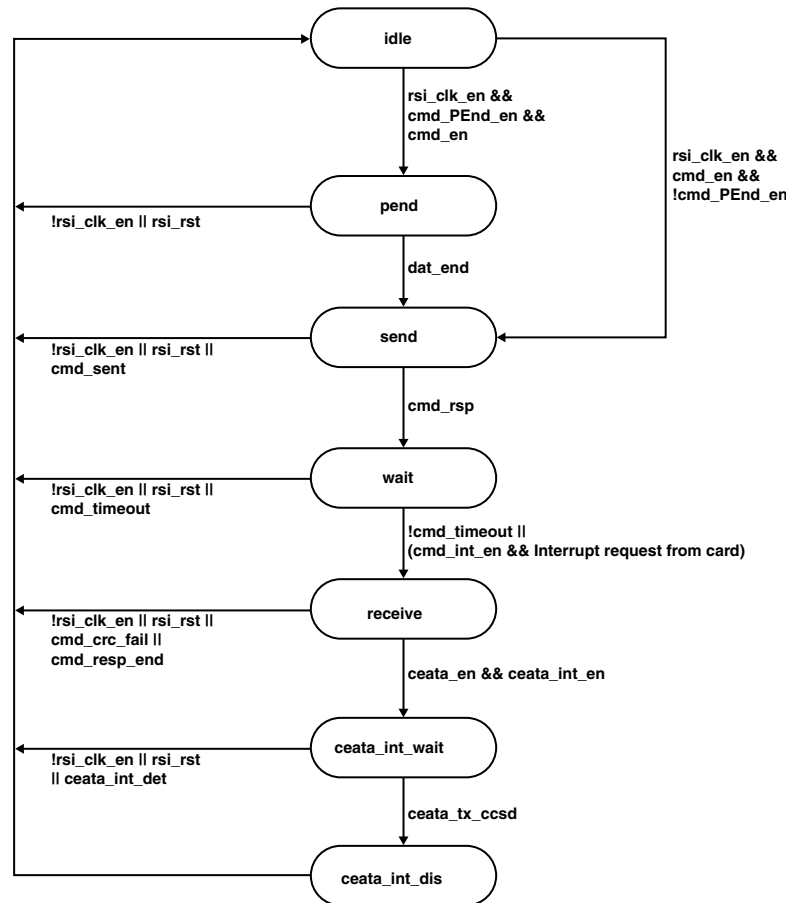


**Figure 24-2:** RSI Command Path State Machine

The command path state machine is responsible for setting and clearing a number of status flags in the RSI\_XFRSTAT register. The following table lists the status flags and exception flags that are affected by the command path state machine.

**Table 24-10:** RSI Command Path Status Flags

| RSI_XFRSTAT/RSI_STAT Flag | Description                     | State Flag Set in |
|---------------------------|---------------------------------|-------------------|
| CMDACT                    | Command transfer is in progress | WAIT_S            |

Table 24-10: RSI Command Path Status Flags (Continued)

| RSI_XFRSTAT/RSI_STAT Flag | Description                                                                        | State Flag Set in |
|---------------------------|------------------------------------------------------------------------------------|-------------------|
| CMDSENT                   | Command without response sent successfully                                         | SEND              |
| CMDTO                     | Response timeout occurred (64 RSI_CLK cycles)                                      | WAIT_S            |
| CMDCRCFAIL                | Response CRC failure                                                               | RECEIVE           |
| RSPEND                    | Response received successfully (and CRC check passed if the CRCDIS bit is cleared) | RECEIVE           |
| CEATAINT                  | CE-ATA command completion signal detected                                          | CEATA_INT_WAIT    |

The command path operates in a half-duplex mode, so that commands and responses can either be sent or received. If the state machine is not in the SEND state, the RSI\_CMD output is in high impedance state.

The following figure describes a typical command and response transfer, the RSI\_CMD signal is sampled by the card and the host on the rising edge of RSI\_CLK.

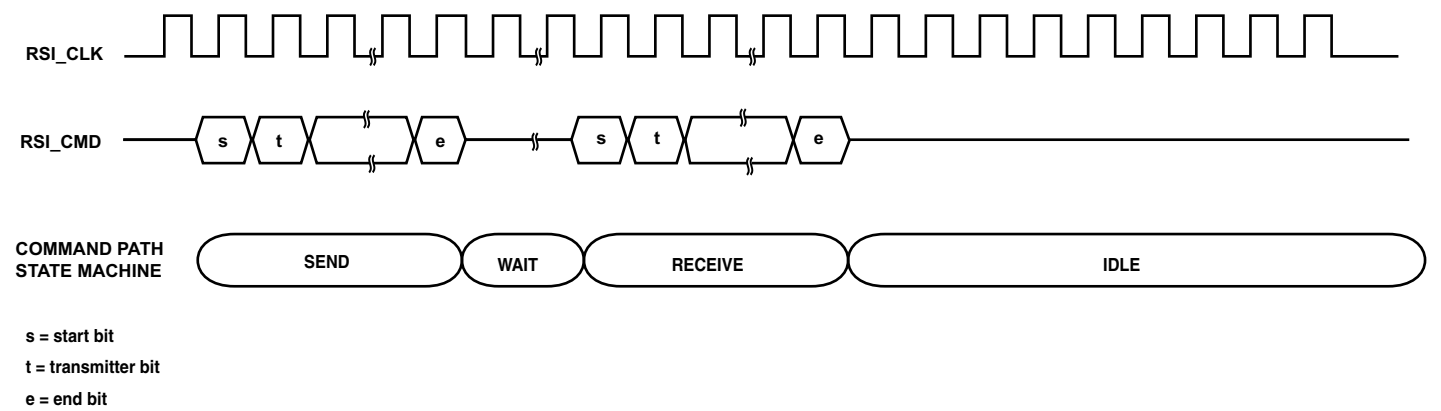


Figure 24-3: RSI Command Transfer

The following sections describes the RSI command path states.

## IDLE State

The command path state machine remains in the IDLE state when it is not active. The command path state machine is enabled and leaves the IDLE state when the EN bit in the RSI\_CMD register is set. The state goes to the PEND state if the PNDEN bit in the RSI\_CMD is set; otherwise it enters the SEND state.



When the command path state machine returns to the IDLE state from another state and the return is not due to RSI being disabled or reset—the state machine remains in the IDLE state for at least eight RSI\_CLK cycles. During this time, the RSI will continue to drive the RSI\_CLK signal even if the PWRSAVE feature is enabled. This allows the card to complete the current operation. If enabled again, the state machine leaves the IDLE state only after the eight RSI\_CLK cycles have passed.

## PEND State

The RSI enters the PEND state if the `PNDEN` bit in the `RSI_CMD` register is set. The state machine remains in the PEND state until it is notified by the data path sub block that a data transfer has completed. This is indicated by the `DATEND` flag being set when `RSI_DATA_CNT` decrements to zero. This mode allows for the automatic transmission of a `STOP_TRANSMISSION` command after reading or writing the required amount of data for stream-based transactions.

The `PNDEN` feature is not functional for block-based transfers and cannot be used to automatically issue the `STOP_TRANSMISSION` command for `MULTIPLE_BLOCK_READ` or `MULTIPLE_BLOCK_WRITE` operations.

## SEND State

During the SEND state, the RSI sets the `CMDACT` flag in the `CMD_XFRSTAT` register to indicate a transfer is in progress. The behavior of the state machine after the command is sent, depends upon whether the command expects a response back from the card.

If no response is expected, the RSI clears the `CMDACT` flag and sets the `CMDSENT` flag to indicate that a command operation without a response has been completed. The state then goes to IDLE.

If a response is expected, the RSI enters the WAIT state.

## WAIT State

In the WAIT state, the RSI waits for a response to be received on the RSI\_CMD signal. Upon entering this state, an internal timer starts. If the response is not received within 64 RSI\_CLK cycles, the `CMDTO` flag is set and the `CMDACT` flag is cleared. The state machine then enters the IDLE state, awaiting the next action.

A response, sent back from the card and indicated by the 0 start bit on the RSI\_CMD signal, transitions the RSI to the RECEIVE state to receive a 48- or 136-bit response.

The WAIT state can also detect card interrupts. This is an optional feature that applies only to MMC cards. This feature is enabled by setting the `IEN` bit within `RSI_CMD`. When `IEN` is set, the timeout timer that is normally started upon entry to the WAIT state is disabled. The RSI remains in this state until a card interrupt is detected. Cards that implement this feature may have functions with a delayed response that is triggered by an internal event in the card. Once the event is triggered the card sends the response. The RSI then detects this start bit of the response and proceeds to the RECEIVE state.

## RECEIVE State

In the RECEIVE state the RSI reads the response on the RSI\_CMD signal from the card. Upon receiving either the short or long response, if the response passes the CRC check, the CMDACT flag is cleared and the RESPEND flag is set. If the CRC check fails, the CMDCRCFAIL flag is set. In either case, the state machine then goes to the IDLE state.

Some CE-ATA commands require additional functionality in this state. A command completion signal must be sent back to the host upon completion of a specific task. For commands that require this functionality, the CEATAMODE and CEATAIEN bits of the RSI\_DATA\_CTL register should be set prior to enabling the command path state machine. After receiving the response, the state machine then enters the CEATA\_INT\_WAIT state.

## CEATA\_INT\_WAIT State

In this state, the RSI waits for the CE-ATA device to issue the command completion signal. This is indicated by the device sending a “0” on the RSI\_CMD signal. Upon detection of the command completion signal, the CMDACT flag is cleared and the CEATAINT flag of the RSI\_STAT0 register is set. Alternatively, the command completion signal of the CE-ATA device can be disabled by the RSI. This is performed by setting the INT\_DIS bit of the RSI\_CEATA register, at which point the state machine enters the CEATA\_INT\_DIS state. The INT\_DIS bit can be set prior to enabling the command path state machine. The CCSD sequence is issued after the response is received rather than having to wait for the response before setting INT\_DIS.

## CEATA\_INT\_DIS State

In this state, the RSI issues the command completion signal disable sequence on the RSI\_CMD signal before it transitions to the IDLE state and clears the CMDACT flag. The command completion signal disable sequence issued is the binary sequence b#00001.

## RSI Command Path CRC

The command CRC generator of the RSI calculates the 7-bit CRC check-sum for all 40 bits preceding the CRC code for both 48-bit commands and 48-bit responses. This includes the start bit, transmitter bit, command index, and command argument (or card status). The 7-bit CRC checksum is calculated for the first 120 bits of the register contents field for the long response format. Note that the start bit, transmitter bit, and the six check bits are not used in the CRC calculation for the long response. The command and response CRC checksum is a 7-bit value that is calculated as follows:

$$\text{CRC}[6:0] = \text{Remainder } \frac{x_7 \times M(x)}{G(x)}$$

With:  $G(x) = x_7 + x_3 + 1$

and for a short response:

$$M(x) = x_{39} \cdot (\text{start bit}) + \dots + x_0 \cdot (\text{last bit before CRC})$$

or for a short response:

$$M(x) = x_{19} \cdot (\text{start bit}) + \dots + x_0 \cdot (\text{last bit before CRC})$$

## RSI Data Interface

Data transfers both to and from the RSI take place over the RSI data bus signals *RSI\_DATA7—0*. The RSI data bus width is configured by the *BUSWID* field of the *RSI\_CTL* register. The default is 1-bit bus mode, where the data is transferred over the *RSI\_DATA0* signal. 4-bit mode or 8-bit mode can be enabled after configuring the card for 4-bit or 8-bit mode of operation, respectively.

The RSI data path state machine operates at *RSI\_CLK* frequency. The state machine leaves the *IDLE* state when the *DATEN* field of the *RSI\_DATA\_CTL* register is set, enabling the data transfer. The state entered upon leaving the *IDLE* state is determined by *DATDIR*. The data path state machine is shown in the following figure.

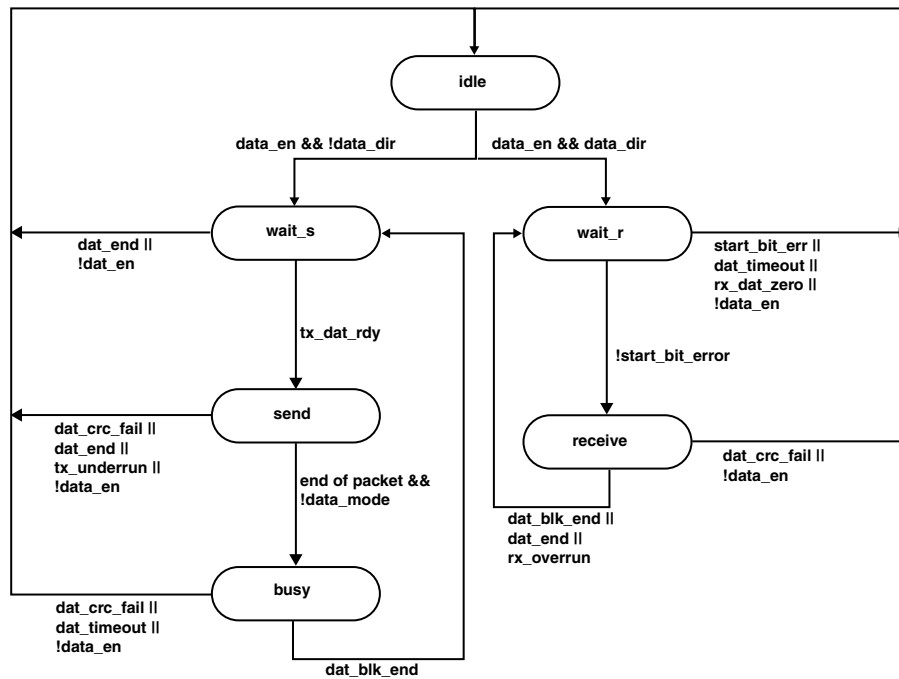


**Figure 24-4:** RSI Data Path State Machine

The data path status flags are shown in the following table.

**Table 24-11:** RSI\_STATUS Flags

| RSI_XFRSTAT Flag | Description                                                                                                                                          | Status Flag Set in                                                                                    |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| TXACT            | Data transmit in progress                                                                                                                            | WAIT_S                                                                                                |
| RXACT            | Data receive in progress                                                                                                                             | WAIT_R                                                                                                |
| DATBLKEND        | Data Block sent successfully and CRC pass<br>token received<br>Data Block received correctly and CRC<br>passed                                       | BUSY (block transfer mode only)<br>RECEIVE (block transfer mode only)                                 |
| DATCRCFAIL       | Data block CRC failed on transmit<br>Data block CRC failed on receive                                                                                | SEND is transmitted data is not a multiple<br>of DATA_BLK_LGTH<br>BUSY if CRC token indicates failure |
| DATTO            | Transmit timeout occurred before card<br>de-asserted busy signal on RSI_DATA0<br>Receive timeout error occurred before<br>start bit of data detected | BUSY<br>WAIT_R                                                                                        |
| DATEND           | All data sent<br>All data received                                                                                                                   | SEND<br>RECEIVE                                                                                       |
| SBITERR          | Start bit not detected on all RSI_DATAx<br>signals                                                                                                   | WAIT_R                                                                                                |
| TXFIFOSTAT       | Transmit FIFO is half empty                                                                                                                          | SEND                                                                                                  |
| TXFIFOFULL       | Transmit FIFO is full                                                                                                                                | SEND                                                                                                  |
| TXFIFOZERO       | Transmit FIFO is empty                                                                                                                               | SEND                                                                                                  |
| TXUNDR           | Transmit FIFO under run error                                                                                                                        | SEND                                                                                                  |
| TXFIFORDY        | Valid data available in the transmit FIFO                                                                                                            | SEND                                                                                                  |
| RXFIFOSTAT       | Receive FIFO is half empty                                                                                                                           | RECEIVE                                                                                               |
| RXFIFOFULL       | Receive FIFO is full                                                                                                                                 | RECEIVE                                                                                               |
| RXFIFOZERO       | Receive FIFO is empty                                                                                                                                | RECEIVE                                                                                               |
| RXOVER           | Receive FIFO over run error                                                                                                                          | RECEIVE                                                                                               |

**Table 24-11:** RSI\_STATUS Flags (Continued)

| RSI_XFRSTAT Flag | Description                                 | Status Flag Set in |
|------------------|---------------------------------------------|--------------------|
| RXFIFORDY        | Valid data is available in the receive FIFO | RECEIVE            |

## RSI Data Transmit Path

The transmit path consists of the *WAIT\_S*, *SEND*, and *BUSY* states. *RSI\_BLK SZ*, *RSI\_DATA\_LEN* and *RSI\_DATA\_TMR* must be configured before enabling the data path state machine with *RSI\_DATA\_CTL*. Upon leaving the *IDLE* state and entering the *WAIT\_S* state, the RSI sets the *TXACT* flag and copies *RSI\_DATA\_LEN* into *RSI\_DATA\_CNT*.

The behavior of the *SEND* state depends on which transfer mode is configured.

- Stream Transfer Mode** - If the RSI is configured for stream transfer mode, it sends data to the card until *RSI\_DATA\_CNT* expires, at which point the *DATEND* flag is set and the state machine returns to the *IDLE* state. Additionally the transition of *RSI\_DATA\_CNT* to zero activates the command path state machine if it is currently in the *PEND* state. If at any point during the stream transfer the transmit FIFO becomes empty and data is not available in the FIFO by the time the next transfer is due to take place, the *TXUNDR* flag is set before returning to the *IDLE* state.
- Block Transfer Mode** - In block transfer mode, *RSI\_BLK SZ* bytes, as specified during the write to *RSI\_DATA\_CTL*, are transmitted. Each byte transferred also decrements *RSI\_DATA\_CNT*. Upon completion of the block transfer, the RSI appends an internally generated 16-bit CRC code and an end bit to the data transferred over the *RSI\_DATAx* signals. The RSI then waits for the card token response on the *RSI\_DATA0* line to indicate whether the data was received correctly by the card or not. If the CRC response token sent by the card indicates the data was received correctly, the *DATBLKEND* flag is set before moving to the *BUSY* state. If the data was not received correctly, the *DATCRCFAIL* flag is set before returning to the *IDLE* state. The decrementing of *RSI\_DATA\_CNT* to zero results in the *DATEND* flag being set. If the total number of bytes transmitted for the current block results in the *RSI\_DATA\_CNT* decrementing to zero and the number of bytes transferred is not equal to *RSI\_BLK SZ*, the transmission stops and the *DATCRCFAIL* flag is set. The data path returns to the *IDLE* state. If at any point during the block transfer the transmit FIFO becomes empty and data is not available in the FIFO by the time the next transfer is due to take place, the *TXUNDR* flag is set before returning to the *IDLE* state.

During the *BUSY* state, the RSI continuously samples *RSI\_DATA0* which at this point is driven low by the card to indicate that the card is busy. When a logic high state is detected, indicating that the card is no longer busy, the state machine returns to the *WAIT\_S* state. It then either returns to *IDLE* if all data has been sent or moves back to the *SEND* state to start another block transfer.

Upon entry to the *BUSY* state, the RSI started decrementing the timeout value specified by *RSI\_DATA\_TMR*. If the RSI timeout counter expires before the *RSI\_DATA0* signal is detected high, the RSI sets the *DATTO* flag and returns to the *IDLE* state.

## RSI Data Receive Path

The receive path consists of the *WAIT\_R* and the *RECEIVE* states. *RSI\_BLKSZ*, *RSI\_DATA\_LEN* and *RSI\_DATA\_TMR* must be configured, before enabling the data path state machine with *RSI\_DATA\_CTL*. Upon leaving the *IDLE* state and entering the *WAIT\_R* state, the RSI sets the *RXACT* flag and copies *RSI\_DATA\_LEN* into *RSI\_DATA\_CNT*. The behavior of the *RECEIVE* state is influenced by the transfer mode.

Once the receive path has entered the *WAIT\_R* state after being enabled for a receive transaction, the RSI starts decrementing the timeout value supplied by the *RSI\_DATA\_TMR*.

If the RSI is configured for a 1-bit data bus, the *DATTO* flag is set if a start bit is not detected on the *RSI\_DATA0* signal before the timeout counter reaches zero. The state machine then returns to the *IDLE* state.

If the RSI is configured for 4-bit bus mode and the start bit is not detected on all four *RSI\_DATAx* signals before the timeout counter expires—the *DATTO* flag is set. The state machine returns to the *IDLE* state. If a start bit is detected on some, but not all, of the *RSI\_DATAx* signals on the same sampled clock cycle, then the *SBITERR* flag is set and the state machine returns to the *IDLE* state. Upon correct detection of the start bit, the state machine moves into the *RECEIVE* state.

The behavior of the *RECEIVE* state differs for stream and block transfers.

- **Stream Transfer Mode** - For stream transfers, received data is packed into bytes and written to the data FIFO. Data is continuously received and written to the data FIFO until *RSI\_DATA\_CNT* decrements to zero. When the counter reaches zero, the remaining data in the shift register is written into the FIFO, the *DATEND* flag is set and the state machine transitions to the *WAIT\_R* state. When the receive FIFO is detected empty, the *RXFIFOZERO* flag is set and the state goes to the *IDLE* state. If the data FIFO becomes full and data has not been read from the FIFO prior to the next byte being written to the FIFO, then the *RXOVER* flag is set. The state goes to the *WAIT\_R* state then into the *IDLE* state.
- **Block Transfer Mode** - In block transfer mode, the received data is packed into bytes and written to the data FIFO. When *RSI\_BLKSZ* bytes have been received, the RSI reads the 16-bit CRC check bits. If the received CRC matches the internally calculated CRC, the *DATBLKEND* flag is set and the state transitions to the *WAIT\_R* state. If the *RSI\_DATA\_CNT* counter expires in alignment with the end of a *RSI\_BLKSZ* block, the *DATEND* and *DATBLKEND* flags are set, and the state goes to the *WAIT\_R* state. When the receive FIFO is detected empty, the *RXFIFOZERO* flag is set and the state goes to the *IDLE* state. If *RSI\_DATA\_CNT* expires before the end of a *RSI\_BLKSZ* block, the *DATCRCFAIL* flag is set. The state goes to the *IDLE* state.

## RSI Data Path CRC

The data CRC generator of the RSI calculates the 16-bit CRC checksum for all bits sent or received for a given block transaction. The data path CRC generator is not enabled for stream based data transfers. For a 1-bit bus configuration, the 16-bit CRC is calculated for all data sent on the *RSI\_DATA0* signal. For a 4-bit-wide data bus, the 16-bit CRC is calculated separately for each *RSI\_DATAx* signal. The data path CRC checksum is a 16-bit value calculated as follows.

$$\text{CRC}[15:0] = \text{Remainder} \frac{x_{16} \cdot M(x)}{G(x)}$$

with:

$$G(x) = x_{16} + x_{12} + x_5 + 1$$

where:

$$M(x) = x_{((8 \times \text{DTX\_BLK\_LGTH}) - 1)} \cdot (\text{first data bit}) + \dots + x_0 \cdot (\text{last data bit})$$

## RSI Data FIFO

The data FIFO is a 32-bit wide, 16-word deep data buffer with transmit and receive logic. The FIFO is configuration depends on the state of the `TXACT` and `RXACT` flags. If `TXACT` is set, the FIFO operates as a transmit FIFO supplying data to the RSI for transfer to the card. If `RXACT` flag is set, the FIFO operates as a receive FIFO, where the RSI writes data received from the card. If neither `TXACT` nor `RXACT` flags are set, then the FIFO is disabled.

When the transmit FIFO is disabled, all the transmit status flags are de-asserted and the transmit read and write pointers are reset. The RSI asserts the `TXACT` flag upon starting a data transfer. During the data transfer the transmit logic maintains the transmit FIFO status flags shown in the following table.

**Table 24-12:** RSI Transmit FIFO Status Flags

| RSI_XFRSTAT Flag | Description                               |
|------------------|-------------------------------------------|
| TXFIFOSTAT       | Transmit FIFO is half empty               |
| TXFIFOFULL       | Transmit FIFO is full                     |
| TXFIFOZERO       | Transmit FIFO is empty                    |
| TXUNDER          | Transmit FIFO under run error             |
| TXFIFORDY        | Valid data available in the transmit FIFO |

When the receive FIFO is disabled, all receive status flags are de-asserted and the receive read and write pointers are reset. The RSI asserts the `RXACT` flag upon starting a data read transaction. During the data transfer, the receive logic maintains the receive FIFO status flags shown in Table "RSI Receive FIFO Status Flags" .

**Table 24-13:** RSI Receive FIFO Status Flags

| RSI_XFRSTAT Flag | Description                              |
|------------------|------------------------------------------|
| RXFIFOSTAT       | Receive FIFO is half empty               |
| RXFIFOFULL       | Receive FIFO is full                     |
| RXFIFOZERO       | Receive FIFO is empty                    |
| RXOVER           | Receive FIFO over run error              |
| RXFIFORDY        | Valid data available in the receive FIFO |

## Card Busy/Ready Detection

Some commands, like CMD6, may assert the busy signal by driving RSI\_DAT0 line low, two cycles after the end bit of the command. RSI\_DAT1-7 lines are driven by the card though their values are not relevant. Software should program the CHKBUSY bit of RSI\_CMD register to enable this busy checking. If this bit is enabled for a particular command, third cycle from the end bit of the command is checked for RSI\_DAT0 going low. If RSI\_DAT0 goes low at this cycle, BUSYMODE bit of RSI\_STAT0 register is set. When the RSI\_DAT0 line goes high, CARDRDY bit of RSI\_STAT0 register is set and BUSYMODE bit is cleared. CARDRDY interrupt is also raised. The CARDRDY bit is a W1C bit and can be cleared by writing 1. If the RSI\_DAT0 doesn't go low at the third cycle from the end bit of the command, card is considered as not busy. BUSYMODE bit of RSI exception status register won't be set, CARDRDY interrupt is raised and CARDRDY bit is set. If the CHKBUSY bit in RSI\_CMD register is not enabled, the RSI\_DAT0 pin is not checked for busy condition.

BUSYMODE bit is also updated during data write operation if the RSI\_DAT0 is pulled low while card is programming the data. But CARDRDY bit and CARDRDY interrupt are not updated for this case.

## SDIO Support

Two additional RSI features implement SDIO functionality.

- Hardware interrupt support over the RSI\_DATA1 pin
- Read wait request over the RSI\_DATA2 pin

SDIO devices may have multiple interrupt sources that are mapped to a single interrupt line. The interrupt is level-sensitive, allowing multiple functions to generate an interrupt simultaneously. Thus the interrupt request will continually be asserted until all sources generating an interrupt are determined and cleared by the RSI.

The sources of the interrupts are found by interrogating the SDIO device. The interrupts are cleared through operations unique to each function.



The SDIO device sends an interrupt request to the RSI by asserting the RSI\_DATA1 signal low. The interrupt status is indicated by the SDIOINT bit of the RSI\_STAT0 register. The status can be configured to interrupt the processor through the SDIOINT bit of the RSI\_IMSK0 register.

When the RSI is configured for 1-bit bus width, the interrupt is generated by the SDIO with no timing constraints since the RSI\_DATA1 signal acts as a dedicated IRQ signal. The RSI should be configured using RSI\_CFG such that pull-up are enabled on all RSI\_DATAx signals. When the RSI sampling RSI\_DATA1 low, the RSI asserts the SDIOINT flag. This flag is asserted until the RSI\_DATA1 signal is sampled high again.

When the RSI is configured for 4-bit bus widths, the RSI\_DATA1 signal is shared between the IRQ signal and the RSI\_DATA1 signal. In this configuration, the interrupt is only be recognized by the RSI within a specific interrupt period.

## RSI Block Diagram

The figure shows the functional blocks within the RSI.

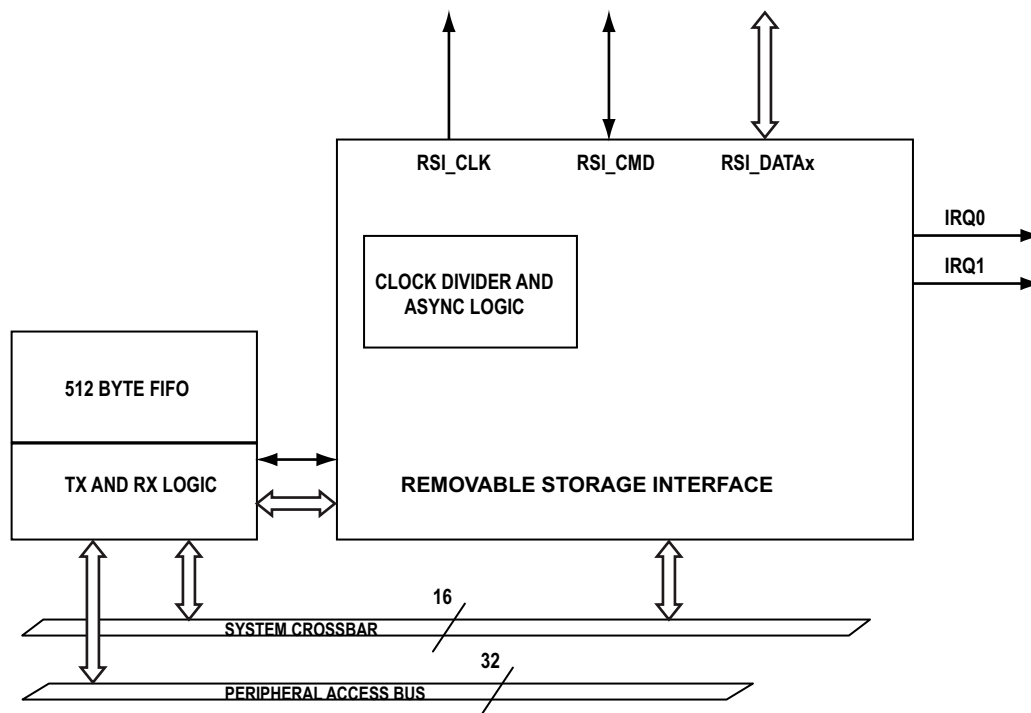


**Figure 24-5:** RSI Block Diagram

## RSI Signal Descriptions

The RSI is a 10 pin interface out of which one pin is used for clock, one pin for command and the rest of the 8 pins are used for data. The [RSI Signal Descriptions \(general\)](#) table "RSI Signal Descriptions (general)" shows a general functional description of various RSI signals.

**Table 24-14:** RSI Signal Descriptions (general)

| Signal Name | Signal Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RSI_CLK     | The clock signal applied to the card from the RSI. All command and data signal transfers are synchronous to this clock. The frequency can vary between zero and the maximum clock frequency. Refer to the respective processor data sheet for the maximum supported clock frequency.                                                                                                                                                                                                                                                |
| RSI_CMD     | A bi-directional command signal used for command transfer and card initialization. The RSI uses this signal to send commands to the cards, and the card uses the signal to send responses back to the RSI. This signal can be configured for both push-pull mode and open-drain mode, but only MMC cards support the open-drain mode. The open-drain mode allows for multiple MMC cards to share data and command signals on the RSI interface and allows for the initialization sequence to take place on all cards <sup>1</sup> . |
| RSI_DATAx   | These are the configurable bi-directional data channels used for all data transfers both to and from the card. The data bus width can be configured as 1-, 4-, or 8-bit.                                                                                                                                                                                                                                                                                                                                                            |

Notes:

1. Although earlier revisions of the MMC specifications allowed for multiple MMC cards to be bused to a single RSI interface, it is strongly recommended that only a single device be interfaced to any given RSI interface. Multiple devices sharing a single command and data bus is now actively discouraged by the device specifications.

The table "RSI Signal Descriptions (MMC and CE-ATA)" and describes the functionality of these signals for MMC and CE-ATA protocols and the table "RSI Signal Descriptions (SD and SDIO)" describes the functionality of these signals for SD and SDIO protocols.

**Table 24-15:** RSI Signal Descriptions (MMC and CE-ATA)

| Signal Name | MMC<br>(1 - bit) | MMC<br>(4 - bit) | MMC<br>(8 - bit) | CE-ATA<br>(4 - bit) | CE-ATA<br>(8 - bit) | Direction      |
|-------------|------------------|------------------|------------------|---------------------|---------------------|----------------|
| RSI_DATA7   | Not Used         | Not Used         | Data7            | Not Used            | Data7               | Bi-directional |
| RSI_DATA6   | Not Used         | Not Used         | Data6            | Not Used            | Data6               | Bi-directional |

**Table 24-15:** RSI Signal Descriptions (MMC and CE-ATA) (Continued)

| Signal Name | MMC<br>(1 - bit)     | MMC<br>(4 - bit)     | MMC<br>(8 - bit)     | CE-ATA<br>(4 - bit)               | CE-ATA<br>(8 - bit)               | Direction      |
|-------------|----------------------|----------------------|----------------------|-----------------------------------|-----------------------------------|----------------|
| RSI_DATA5   | Not Used             | Not Used             | Data5                | Not Used                          | Data5                             | Bi-directional |
| RSI_DATA4   | Not Used             | Not Used             | Data4                | Not Used                          | Data4                             | Bi-directional |
| RSI_DATA3   | Not Used             | Data3                | Data3                | Data3                             | Data3                             | Bi-directional |
| RSI_DATA2   | Not Used             | Data2                | Data2                | Data2                             | Data2                             | Bi-directional |
| RSI_DATA1   | Not Used             | Data1                | Data1                | Data1                             | Data1                             | Bi-directional |
| RSI_DATA0   | Data0                | Data0                | Data0                | Data0                             | Data0                             | Bi-directional |
| RSI_CMD     | Command/<br>Response | Command/<br>Response | Command/<br>Response | Command/<br>Response/<br>CCS/CCSD | Command/<br>Response/<br>CCS/CCSD | Bi-directional |
| RSI_CLK     | Clock                | Clock                | Clock                | Clock                             | Clock                             | Output         |

**Table 24-16:** RSI Signal Descriptions (SD and SDIO)

| Signal Name | SD<br>(1 - bit) | SD<br>(4 - bit) | SDIO<br>(1 - bit) | SDIO<br>(4 - bit) | Direction      |  |
|-------------|-----------------|-----------------|-------------------|-------------------|----------------|--|
| RSI_DATA7   | Not Used        | Not Used        | Not Used          | Not Used          | Bi-directional |  |
| RSI_DATA6   | Not Used        | Not Used        | Not Used          | Not Used          | Bi-directional |  |
| RSI_DATA5   | Not Used        | Not Used        | Not Used          | Not Used          | Bi-directional |  |
| RSI_DATA4   | Not Used        | Not Used        | Not Used          | Not Used          | Bi-directional |  |

**Table 24-16:** RSI Signal Descriptions (SD and SDIO) (Continued)

| Signal Name | SD<br>(1 - bit)          | SD<br>(4 - bit)       | SDIO<br>(1 - bit)        | SDIO<br>(4 - bit)                 | Direction      |  |
|-------------|--------------------------|-----------------------|--------------------------|-----------------------------------|----------------|--|
| RSI_DATA3   | Not Used/<br>Card Detect | Data3/<br>Card Detect | Not Used/<br>Card Detect | Data3/<br>Card Detect             | Bi-directional |  |
| RSI_DATA2   | Not Used                 | Data2                 | Read Wait                | Data2/<br>Read Wait               | Bi-directional |  |
| RSI_DATA1   | Not Used                 | Data1                 | Interrupt                | Data1/<br>Interrupt               | Bi-directional |  |
| RSI_DATA0   | Data0                    | Data0                 | Data0                    | Data0                             | Bi-directional |  |
| RSI_CMD     | Command/<br>Response     | Command/<br>Response  | Command/<br>Response     | Command/<br>Response/<br>CCS/CCSD | Bi-directional |  |
| RSI_CLK     | Clock                    | Clock                 | Clock                    | Clock                             | Output         |  |

## RSI Architectural Concepts

The following sections describe the functions and features of the RSI controller as well as the MMC, SD, SDIO, and CE-ATA protocols. For detailed information on timing parameters and protocol requirements, refer to the corresponding processor data sheet and the following standards and specifications.

- MMCA System Specification
- JESD84 series of JEDEC standards
- SD Specifications Part 1 Physical Layer Specification
- SD Specifications Part 1 Physical Layer Simplified Specification
- SD Specifications Part E1 SDIO Specification

Communication is through a master and slave configuration, where the RSI is the master and the card is the slave device. The RSI communicates with the device using a message-based bus protocol in which the host sends commands serially using the *RSI\_CMD* signal. Some commands require that the card provide a response back to the host. This response is also sent serially by the *RSI\_CMD* signal.

Data transfers, both to and from the card, occur using the *RSI\_DATAx* signals. The number of data lines used for the data transfer can be configured to 1, 4, or 8 using *RSI\_DATA0*, *RSI\_DATA3—0*, or *RSI\_DATA7—0*. All *RSI\_CMD* and *RSI\_DATAx* transfers are synchronous with *RSI\_CLK*.

Cyclic redundancy codes (CRC) are used to protect commands, responses and data transfers from transmission errors. A CRC7 code is generated for every command sent by the host and for almost every response returned by the card on the RSI\_CMD signal. A CRC16 code is used on the RSI\_DATAx signals to protect block data transfers. In 4- and 8-bit bus configurations, CRC16 is calculated for each individual data signal.

## RSI Operating Modes

The following sections describe the functionality of the RSI module in various modes of operation.

### Card Identification Mode

When a device connected to the RSI is first powered and detected by the host or has been reset, the device must first be identified and initialized by the host. The software determines whether the device is compatible with the RSI controller and the implemented software drivers. This phase in the procedure is known commonly as the card identification mode.

When a device is in card identification mode, the host may be required to perform the following actions.

- Reset the device
- Validate the device operating voltage range
- Identify the device type
- Assign/request a relative card address (RCA)

All communications between the host and card during the card identification phase occur using the RSI\_CMD signal. The maximum clock frequency during the identification phase is typically far lower than the maximum data transfer frequency for the card.

### Data Transfer Mode

The card enters stand-by state, known as the data transfer mode, when it has been assigned an RCA. Data transfers can only take place when the device has entered the data transfer mode.

Once the device is in data transfer mode, communication takes place through the RSI\_CMD and the RSI\_DATAx signals. The card is further interrogated to identify bus widths, maximum clock frequency, and the device capacity. At this point the bus width can be altered and the clock frequency can be increased.

Data may be written to the device or read from the device using the following two methods.

- Stream reads and writes
- Block reads and writes

Stream transfers produce a continuous stream of data until the RSI stops the transfer by setting a specific command.

For stream read and write operations, additional maximum operating frequency limitations may be imposed by the device. Stream write operations may also have restrictions that are dependent upon writable block boundaries.

Block based transfers result in a block of a pre-configured size being transferred. The block size depends on the device and is obtained by reading registers contained on the device during the device detection procedure.

The data transfer between the RSI block and the processor's internal memory can be performed in two ways as described below.

## DMA Data Transfers

The RSI block has a dedicated DMA channel assigned. This DMA channel can be configured either to transmit/receive the data from/to the RSI data FIFO without core intervention.

## Core Data Transfers

If the RSI DMA channel is disabled, the RSI transmit/receive FIFO may be written or read by the processor core as a memory-mapped register (RSI\_FIFO). In order to avoid FIFO overflow or underflow, the core should access the FIFO registers in one of the two following ways.

1. Unmask the transmit FIFO empty (TXFIFOZERO) or receive FIFO full (RXFIFOFULL) interrupt in the RSI\_XFR\_IMSK0/1 register. Write the word to be set out to the transmit FIFO or read the received word from the receive FIFO inside the interrupt service routine.
2. Poll the transmit FIFO empty (TXFIFOZERO) or receive FIFO full (RXFIFOFULL) status bit in the RSI\_XFR\_STAT register. Write the word to be set out to the transmit FIFO or read the received word from the receive FIFO after the corresponding status bit is detected as set.

## Boot Mode

MultiMedia Cards (MMC) based on the MMCA specifications version 4.3 or later support a special mode known as boot mode. In this mode, the MMC host can read boot data from the slave MMC device by keeping RSI\_CMD line low after power-on, or sending CMD0 with argument 0xFFFFFFFF (optional for slave), before issuing CMD1. The data can be read from either a dedicated boot area or user area.

## Normal Boot Mode

Before enabling the RSI boot, boot timing counters (SETUP and HOLD) in the RSI\_BOOT\_TCNTR register and boot mode type (MMCBMODE) in the RSI\_CFG register should be programmed. Otherwise RSI will use the default values of 74 and 56 RSI\_CLK cycles. DATA\_TMR register should be programmed to act as boot data timeout register. If the BACKEN bit of the RSI\_CFG register is set, boot acknowledge timeout (RSI\_BACK\_TOUT) register should also be programmed. For normal boot mode, MMCBMODE bit in RSI\_CFG register is written 0.

When the RSI block is enabled for normal boot mode, by setting the MMCBEN bit of the RSI\_CFG register, following tasks are done at the same cycle.

1. RSI\_CMD line is pulled low.
2. Boot setup counter (RSI\_BOOT\_TCNTR) starts decrementing.
3. Boot acknowledge timeout (RSI\_BACK\_TOUT) counter starts incrementing if the BACKEN bit is set.
4. Boot data timeout counter (DATA\_TMR) starts incrementing.

When the boot setup counter expires, BSETUPEXP interrupt is raised and also BSETUPEXP bit of RSI\_STAT0 register will be set. BSETUPEXP interrupt indicates that card has entered the boot state. Core can clear this status bit by writing 1 to BSETUPEXP bit of RSI\_STAT0 register. Clearing this bit also clears the BSETUPEXP interrupt line.

If BACKEN bit is set in RSI\_CFG register, the RSI goes to boot state and expects an acknowledgement to be returned by the slave within an interval as programmed in RSI\_BACK\_TOUT counter. If this acknowledge is not received before the counter reaches this value, BACKTO interrupt is raised and also BACKTO bit of RSI\_STAT0 register is set. If the acknowledge is received, but is not “010”, BACKBAD interrupt is raised and also BACKBAD of RSI\_STAT0 register is set. In both the above cases RSI goes back to IDLE state. Core can clear the status bits by writing 1 to RSI\_STAT0 register. Core should terminate the boot operation by pulling RSI\_CMD line high by writing 0 to MMCBEN bit of RSI\_CFG register. Now boot hold counter (RSI\_BOOT\_TCNTR) starts decrementing. Once the counter expires, BHOLDEXP interrupt is raised and BHOLDEXP bit of RSI\_STAT0 register will be set. Core can clear this status bit by writing 1 to the same bit. From here RSI can start normal operation (non boot operation even though boot operation failed).

If the acknowledge is received before the RSI\_BACK\_TOUT counter expires, RSI enters WAIT\_R state once the data transfer is enabled by writing to RSI\_DATA\_CTL register. RSI waits for the start bit of the data.

If BACKEN bit is 0 in the RSI\_CFG register, then no acknowledgement is expected and RSI is in WAIT\_R state once the data transfer is enabled by writing to RSI\_DATA\_CTL register.

Slave should start sending the boot data before boot data timeout (DATA\_TMR) counter expires. If the counter expires while state machine is still in WAIT\_R state, BDATTO interrupt is raised and also BDATTO bit is set. Core can clear this bit by writing 1 to bit 10 of the bit of the RSI\_STAT0 register. RSI goes to IDLE state after timeout.

Once the entire boot data is received, core can terminate the boot operation by pulling RSI\_CMD line high by writing 0 to MMCBEN bit of RSI\_CFG register. Now boot hold counter (RSI\_BOOT\_TCNTR) starts decre-

menting. Once the counter expires, BHOLDEXP interrupt is raised and also BHOLDEXP bit of RSI\_STAT0 register will be set. Core can clear this status bit by writing 1 to the same bit. From here, RSI can start normal operation (non boot operation even though boot operation failed).

Now the slave is ready for CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

## Alternate Boot Mode

Before enabling the RSI boot, boot timing counters (SETUP and HOLD) in the RSI\_BOOT\_TCNTR register and boot mode type (MMCBMODE) in the RSI\_CFG register should be programmed. Otherwise RSI uses the default values of 74 and 56 RSI\_CLK cycles. DATA\_TMR register should be programmed to act as boot data timeout register. If the BACKEN bit of the RSI\_CFG register is set, boot acknowledge timeout (RSI\_BACK\_TOUT) register should also be programmed. For alternate boot mode, MMCBMODE bit in RSI\_CFG register is written 1.

When the RSI block is enabled for alternate boot mode, by setting the MMCBEN bit of the RSI\_CFG register, Boot setup counter (RSI\_BOOT\_TCNTR) starts decrementing. When this counter expires, BSETUPEXP interrupt is raised and also status bit BSETUPEXP will be set in the RSI\_STAT0 register. Core can clear this status bit by writing 1 to the same bit. Only after getting BSETUPEXP interrupt, CMD0 with argument 0xFFFFFFFFFA should be sent to slave to start the boot operation. When the end bit of this CMD0 is reached on the CMD line, boot acknowledge timeout counter (RSI\_BACK\_TOUT) and boot data timeout counter (DATA\_TMR) start incrementing from zero.

If BACKEN bit is set in RSI\_CFG register, the RSI goes to boot state and expects an acknowledgement to be returned by the slave within an interval as programmed in RSI\_BACK\_TOUT counter. If this acknowledge is not received before the counter expires, BACKTO interrupt is raised and also status bit BACKTO is set in RSI\_STAT0 register. If the acknowledge is received, but is not "010", BACKBAD interrupt is raised and also status bit BACKBAD is set. In both the above cases RSI goes back to IDLE state. Core can clear the status bits by writing 1 to respective bits of RSI\_STAT0 register.

Core can terminate the boot operation first by clearing the MMCBEN bit of RSI\_CFG register and then sending CMD0 command. Now boot hold counter (RSI\_BOOT\_TCNTR) starts decrementing. Once the counter expires, BHOLDEXP interrupt is raised and also status bit BHOLDEXP will be set. Core can clear this status bit by writing 1 to RSI\_STAT0 register. From here, RSI can start normal operation (non boot operation even though boot operation failed).

If the acknowledge is received before the RSI\_BACK\_TOUT counter expires, RSI enters WAIT\_R state once the data transfer is enabled by writing to RSI\_DATA\_CTL register. RSI waits for the start bit of the data.

If BACKEN bit is 0 in the RSI\_CFG register, then no acknowledgement is expected and RSI is in WAIT\_R state once the data transfer is enabled by writing to RSI\_DATA\_CTL register.

Slave should start sending the boot data before boot data timeout counter (DATA\_TMR) expires. If the counter expires while state machine is still in WAIT\_R state, BDATTO interrupt is raised and also BDATTO flag is set. Core can clear this bit by writing 1 to the same bit. RSI goes to IDLE state after timeout.



When the entire boot data is received, core can terminate the boot operation by clearing `MMCBEN` bit of `RSI_CFG` register and sending `CMD0` command. Now boot hold counter (`RSI_BOOT_TCNT`) starts decrementing. Once the counter expires, `BHOLDEXP` interrupt is raised and also status bit `BHOLDEXP` will be set. Core can clear this status bit by writing 1 to `BHOLDEXP` bit `RSI_STAT0` register. From here, RSI can start normal operation (non boot operation even though boot operation failed).

Now the slave is ready for `CMD1` operation and the master needs to start a normal MMC initialization sequence by sending `CMD1`.

Note: If Boot is abruptly terminated for whatever reason, booting cannot be retried without power off and power on to the MMC card.

## Sleep Mode

MultiMedia Cards (MMC) based on the MMCA specification version 4.3 or later support this mode. A card may be switched between a sleep state and a standby state by `SLEEP/WAKEUP` command (`CMD5`). In the sleep state, the power consumption of the memory device is minimized. The `SLEEP` command is used to initiate the state transition from standby state to sleep state. The memory device indicates the transition phase busy by pulling the `RSI_DATA0` line low. The sleep state is reached when the memory device stops pulling down the `RSI_DATA0` line low (`RSI_DATA0` goes high).

The `WAKEUP` command is used to initiate the state transition from sleep state to standby state. The memory device indicates the transition phase busy by pulling down the `RSI_DATA0` line low. The standby state is reached when the memory device stops pulling down the `RSI_DATA0` line low (`RSI_DATA0` goes high).

When in standby state, card is allowed to go to sleep mode by sending the `CMD5` with bit 15 set in argument. `SLP_WKUP_TOUT` counter starts decrementing at the end bit of this command. RSI block is now waiting for a rising edge on the `RSI_DATA0` line. If the rising edge is not detected on the `RSI_DATA0` line before the counter expires, `SLPWKPTOUT` interrupt is raised. Also `SLPWKPTOUT` bit of `RSI_STAT0` register is set.

If a rising edge is detected on `RSI_DATA0` pin before the `SLP_WKUP_TOUT` counter expires, `SLPDONE` interrupt is raised and also `SLPDONE` bit and `SLPMODE` bit of `RSI_STAT0` register are set. This indicates that card has successfully entered sleep state. `SLPDONE` bit is a W1C bit and can be cleared by writing 1. `SLPMODE` bit is a read-only bit and can not be cleared by writing 1. Now the `Vcc` power supply for the card may be switched off. The `Vcc` supply have to be ramped back up before the state transition from the sleep state to standby state initiated by sending the `WAKEUP` command.

When the `WAKEUP` command is issued by sending `CMD5` with bit 15 as 0 in argument, `SLP_WKUP_TOUT` counter starts decrementing. If a rising edge on the `RSI_DATA0` line is not detected before the counter expires, `SLPWKPTOUT` interrupt is raised. Also `SLPWKPTOUT` bit of `RSI_STAT0` register is set.

If a rising edge is detected on `RSI_DATA0` pin before the counter expires, `WKPDONE` interrupt is raised and `WKPDONE` bit of `RSI_STAT0` register is set and `SLPMODE` bit of `RSI_STAT0` register is cleared. This indicates that card has successfully entered standby state. Now normal operation can be resumed.

A SLEEP command is recognized only if the card is in the standby state and the WAKEUP command is recognized only when the card is in the sleep state.

## RSI Data Transfer Modes

{Editor's note: This topic is an optional, example topic. If it does not apply to the module, deleted it. If it does apply to the module, please keep it, retain this name for it, and have it appear in this location in the chapter.}

## RSI Signal Modes

{Editor's note: This topic is an optional, example topic. If it does not apply to the module, deleted it. If it does apply to the module, please keep it, retain this name for it, and have it appear in this location in the chapter.}

## RSI Event Control

This section provides details about various RSI interrupt, status, and error signals.

## RSI Interrupt Signals

The RSI has two interrupt signals (IRQ0 and IRQ1) which connect the module to the System Event Controller (SEC).

## RSI Status and Error Signals

The RSI block has 22 individual status bits in the `RSI_XFRSTAT` register that can be configured to generate an interrupt. The status bits can be mapped to either of the two interrupts IRQ0 or IRQ1. This allows for greater flexibility in system configuration. To generate an interrupt on IRQ0, the interrupt should be enabled by setting the corresponding bit in the `RSI_XFR_IMSK0` register. Interrupts that are required to be generated on IRQ1 are enabled by setting the corresponding bit in the `RSI_XFR_IMSK1` register.

In addition the `RSI_XFRSTAT` register, each of the flags in the `RSI_STAT0` register are also capable of generating an interrupt. Interrupts for the `RSI_STAT0` flags are enabled by setting the corresponding bit in the `RSI_IMSK0` register. These interrupts are sent to the SEC through IRQ0 only.

## RSI Programming Model

The following sections describe the RSI programming model for various RSI operating modes.

## Card Identification

### SD Card Identification Procedure

1. Issue the IDLE command to the card using the `RSI_CMD` register.
2. Issue the SEND\_IF command through the `RSI_CMD` register, supplying the host supply voltage and a check pattern via the `RSI_ARG` register. The command expects an R7 response type. If a valid response with a compatible voltage range and matching check pattern is received, the card is compliant with SD vSDHon 2.00 or later. If a response is received with an incompatible voltage range the card cannot be used. If no response is received at all, as indicated by the `CMDTO` flag on the `RSI_XFRSTAT` register, go to step 5.
3. Issue the `RSI_SEND_OP_COND` command through the `RSI_CMD` register, supplying the voltage window supported and whether the host supports high capacity cards using the `RSI_ARG` register. The RSI expects an R3 response to this command. The RSI can reject the card at this point if the voltage range is not compatible. If the card returns a response indicating that it is busy, resend the `RSI_SEND_OP_CMD` until the card indicates it is ready. If the host does not support the high capacity mode as indicated by setting the HCS bit of the argument to 0—a high capacity card never clears the busy status bit. The card should be identified within one second. If in that time the card is still bus, or no valid responses have been received, the card is rejected.
4. If the host supports high-capacity cards, verify whether the response in the `RSI_RESP0` register indicates that the card capacity status (CCS) bit is set. If CCS is set, an SD VeSDHon 2.00 or later high capacity SD memory card is present—proceed to step 6.
5. If the CCS bit is cleared, then the card is an SD VeSDHon 2.00 or later standard capacity memory card— proceed to step 6. Issue the `RSI_SEND_OP_COND` command via the `RSI_CMD` register, supplying the voltage window supported and with the high-capacity support (HCS) bit set to 0 via the `RSI_ARG` register. The RSI expects an R3 response to this command, at which time the card can be rejected if the voltage range is not compatible. If the card returns a response indicating that it is busy, resend the `RSI_SEND_OP_CMD` until the card indicates that it is ready. The card should be identified within 1 second. If in that time frame the card is still busy or no valid responses have been received, the card should be rejected. Once the response indicates that the card is ready, the card type has been identified as an SD Version 1.x standard-capacity memory card.
6. Issue the `ALL_SEND_CID` command to which an R2 response type is expected. This will result in the card sending the 128-bit card identification (CID) register and transitioning from ready to identification mode.
7. Issue the `SEND_RELATIVE_ADDR` command to which an R6 response type is expected. This will result in the card issuing a new relative address which must be used in order to select the card in the future for data transfers. The card will then move into standby mode completing the identification procedure.

## MMC Card Identification Procedure

1. Issue the IDLE command to the card via the `RSI_CMD` register.
2. Issue the `SEND_OP_COND` command to the card via the `RSI_CMD` register, supplying the operating voltage window that the host is compatible with and the access mode that the host supports (byte or sector) via the `RSI_ARG` register. The RSI expects an R3 type response. This allows the host to reject the card if it is not compatible with the supply voltage or if the access mode is not supported by the host software. If the card returns an indication that it is busy, repeat this step until the card is either rejected or not busy.
3. Issue the `ALL_SEND_CID` command via the `RSI_CMD` register. The RSI expects an R2 response to this command. This will result in the card sending the 128-bit card identification (CID) register and transitioning from ready to identification mode.
4. Issue the `SET_RELATIVE_ADDR` command, providing a 16-bit relative card address (RCA) via the `RSI_ARG` register that will get assigned to the card. An R1 response type is expected for this command. This will result in the card being assigned with the provided RCA, which must be used in order to select the card in the future for data transfers. The card will then move into standby mode, completing the identification procedure.

## Data transfer

The following section describe how to program the RSI block for various data transfer scenarios in both core and DMA modes.

### Single Block Write

Block write operations typically consist of 512 bytes of data per block. If the card is found to support other block lengths or the default block length as specified in the CID register is not 512, the block length of the RSI must be configured accordingly. The block length of the card and the block length of the RSI must be configured for the same block size at all times. The block length of the RSI is configured via the `RSI_BLKSZ` register.

NOTE: It is important to pay attention as to when the data path state machine is enabled and when data is written to the FIFO for transfer to the card. Write transactions require that data be written after the response has completed for the `WRITE_BLOCK` command. If the data path state machine is enabled prior to sending the `WRITE_BLOCK` command, data must not be written to the transmit FIFO via the DMA or core until after the response has been received as indicated by the `RESPEND` flag of the `RSI_XFRSTAT` register. Failure to adhere to this procedure can result in data being written to the card in violation to the block write timing parameters, resulting in a data CRC failure.

## Using Core

1. Write the `RSI_ARG` register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.
3. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in `RSI_RESP0`.
5. Write the number of bytes to be transferred to the `RSI_DATA_LEN` register. This will be 512 bytes for a single block.
6. Write the appropriate timeout value for a write operation to the `RSI_DATA_TMR` register.
7. Write the destination start address to the `RSI_ARG` register. The supplied address must be aligned to a 512 byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or sector-addressable MMC card.
8. Write the `WRITE_BLOCK` command to the `RSI_CMD` register, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
9. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
10. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN` should also be set to enable the data path state machine. All other fields of the `RSI_DATA_CTL` register should be zero.
11. Write data to the `RSI_FIFO` register until the FIFO becomes full as indicated by the `TXFIFOFULL` flag of the `RSI_XFRSTAT` register. Continue to write data to the FIFO as long as the FIFO is not full or write data in blocks of eight 32-bit words if polling on the `TXFIFOSTAT` bit indicating the transmit FIFO is half empty. Continue until all 128 32-bit words (512 bytes) have been transferred.
12. Wait for the card to respond with the CRC token by waiting for the `DATBLKEND` flag to be set. `DATEND` will also be set at this time if the `RSI_DATA_LEN` register was set to 512 bytes in step 5.
13. Clear the `DATBLKEND` and `DATEND` flags via the `RSI_XFRSTAT_CLR` register.

## Using DMA

1. Write the `RSI_ARG` register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.
3. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register, and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in `RSI_RESP0`. Configure the DMA channel assigned to the RSI controller.
5. Write `DMAx_ADDRSTART` with the address of the first byte of data to be written to the card. The `DMAx_XCNT` register should be set to 128, and the `DMAx_XMOD` register to 4. The `DMAx_CFG` register should be set for DMA enable (a word size of 32-bits).
6. Once the DMA channel has been configured and enabled, write the number of bytes to be transferred to the `RSI_DATA_LEN` register. This will be 512 bytes for a single block.
7. Write the appropriate timeout value for a write operation to the `RSI_DATA_TMR` register.
8. Write the destination start address to the `RSI_ARG` register. The address supplied must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or sector-addressable MMC card.
9. Write the `WRITE_BLOCK` command to `RSI_CMD`, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
10. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register, and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
11. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN` and `DMAEN` should also be set to enable the data path state machine and to allow the DMA controller to access the transmit FIFO. All other fields of the `RSI_DATA_CTL` register should be zero.
12. Wait for the card to respond with the CRC token by waiting for the `DATBLKEND` flag to be set. `DATEND` will also be set at this point if the `RSI_DATA_LEN` register was set to 512 bytes in step 5.
13. Clear the `DATBLKEND` and `DATEND` flags via the `RSI_XFRSTAT_CLR` register. Also clear the `IRQDONE` bit of the `DMAx_STAT` register, if applicable.

## Single Block Read

Block read operations typically consist of 512 bytes of data per block. If the card is found to support other block lengths or the default block length as specified in the CID register is not 512, the block length of the RSI must be configured accordingly. The block length of the card and the block length of the RSI must be

configured for the same block size at all times. The block length of the RSI is configured via the `RSI_BLKSZ` register.

NOTE: It is important to pay attention as to when the data path state machine is enabled and when data is read from the receive FIFO for data transfers from the card to the RSI. Read transactions can occur on the `RSI_DATAx` signals prior to the response of the command being received. It is therefore advisable to enable the data path state machine, and DMA controller if being used, either:

- Prior to issuing a command that involves a data read packet
- Immediately after the command has been issued but prior to pending on the `RESPEND` flag

NOTE: If the core is being used to read the receive FIFO, it is advised not to pend on the `RESPEND` flag. It is possible for data to be driven on the `RSI_DATAx` signals two `RSI_CLK` cycles after the end bit of the command. At minimum, an additional 48 `RSI_CLK` cycles will pass before the response is received, during which time the receive buffer may potentially have received 24 bytes of data on a 4-bit bus and will be approaching the half full state. Software should ensure that the receive buffer does not become full prior to data being read from the receive FIFO.

## Using Core

1. Write the `RSI_ARG` register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.
3. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in `RSI_RESP0`.
5. Write the number of bytes to be transferred to the `RSI_DATA_LEN` register. This will be 512 bytes for a single block.
6. Write the appropriate timeout value for a read operation to the `RSI_DATA_TMR` register.
7. Write the destination start address to the `RSI_ARG` register. The address supplied must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or sector-addressable MMC card.
8. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN` and `DATDIR` should also be set to enable the data path state machine and indicate the transfer direction is from card to controller. All other fields of the `RSI_DATA_CTL` register should be zero.
9. Write the `READ_SINGLE_BLOCK` command to the `RSI_CMD` register, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.

10. In order to meet some timing restrictions related to block read operations, it is advisable to not wait for the `RESPEND` indication within the `RSI_XFRSTAT` register but instead move immediately on the next step. This is due to the card being able to send data before a response can be completed on the `RSI_CMD` signal, moving immediately onto step 11 will ensure a receive FIFO overflow does not occur.
11. Poll the `RXFIFORDY` bit or the `RXDATZERO` bit of `RSI_XFRSTAT` indicating the receive FIFO has data available, or the receive FIFO is empty. As long as the receive FIFO is not empty, read data from the `RSI_FIFO` register until all 512 bytes have been read.
12. Once all bytes have been read, wait for the `DATBLKEND` flag to indicate that the data was received correctly and passed the CRC check. The `DATEND` flag may also be set, depending on the value written to `RSI_DATA_LEN`.
13. Clear the `DATBLKEND` and `DATEND` flags via the `RSI_XFRSTAT_CLR` register.

## Using DMA

1. Write the `RSI_ARG` register with the card's RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.
3. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register, and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in `RSI_RESP0`.
5. Configure the DMA channel assigned to the RSI controller. Write `DMAx_ADDRSTART` with the address of the first byte of where the received data is to be stored. The `DMAx_XCNT` register should be set to 128 and the `DMAx_XMOD` register to 4. The `DMAx_CFG` register should be set for DMA enable (a word size of 32-bits and direction set to memory write).
6. Write the number of bytes to be transferred to the `RSI_DATA_LEN` register. This will be 512 bytes for a single block.
7. Write the appropriate timeout value for a read operation to the `RSI_DATA_TMR` register.
8. Write the source start address to the `RSI_ARG` register. The supplied address must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or sector-addressable MMC card.
9. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN`, `DATDIR`, and `DMAEN` should also be set to enable the data path state machine, set the transfer direction from card to controller and allow the DMA controller access to the receive FIFO. All other fields of the `RSI_DATA_CTL` register should be zero.



10. Write the READ\_SINGLE\_BLOCK command to the RSI\_CMD register, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1.
11. Unlike core accesses, it is safe to poll on RESPEND indication within the RSI\_XFRSTAT register and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register. The DMA controller enabled in step 5 will ensure any data sent to the receive FIFO prior to the RESPEND flag being set is received correctly.
12. Wait for the DATBLKEND flag to indicate that the data was received correctly and passed the CRC check. The DATEND flag may also be set, depending on the value written to RSI\_DATA\_LEN.
13. Clear the DATBLKEND and DATEND flags via the RSI\_XFRSTAT\_CLR register. Also clear the IRQDONE bit of the DMAX\_STAT register, if applicable.

## Multiple Block Write

Block write operations typically consist of 512 bytes of data per block. If the card is found to support other block lengths or the default block length as specified in the CID register is not 512, the block length of the RSI must be configured accordingly. The block length of the card and the block length of the RSI must be configured for the same block size at all times. The block length of the RSI is configured via the RSI\_BLK SZ field of the RSI\_DATA\_CTL register.

## Using Core

1. Write the RSI\_ARG register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the RSI\_ARG register.
2. Write the RSI\_CMD register with the SELECT/DESELECT\_CARD command, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1b.
3. Wait for the RESPEND indication within the RSI\_XFRSTAT register and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in RSI\_RESP0.
5. Write the number of bytes to be transferred to the RSI\_DATA\_LEN register. For example, write 4096 to write eight blocks of 512 bytes.
6. Write the appropriate timeout value for a write operation to the RSI\_DATA\_TMR register.
7. Write the destination start address to the RSI\_ARG register. The supplied address must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or a sector-addressable MMC card.
8. Write the WRITE\_MULTIPLE\_BLOCK command to RSI\_CMD, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1.

9. Wait for the RESPEND indication within the RSI\_XFRSTAT register, and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register.
10. Enable the data path state machine by writing to the RSI\_DATA\_CTL register with RSI\_BLK SZ set to 512 for a 512-byte block. DATEN should also be set to enable the data path state machine. All other fields of the RSI\_DATA\_CTL register should be zero.
11. Write data to the RSI\_FIFO register until the FIFO becomes full as indicated by the TXFIFOFULL flag of the RSI\_XFRSTAT register. Continue to write data to the FIFO as long as the FIFO is not full or write data in blocks of eight 32-bit words if polling on the TXFIFOSTAT bit indicating the transmit FIFO is half empty. Continue until all 128 32-bit words (512 bytes) have been transferred.
12. Wait for the card to respond with the CRC token by waiting for the DATBLKEND flag to be set.
13. Clear the DATBLKEND flag.
14. Repeat steps 11 to 13 for the number of blocks to be transferred or until DATEND flag is set.
15. Write the RSI\_CMD register with the STOP\_TRANSMISSION command, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1. 16.
16. Clear the DATEND and RESPEND flags via the RSI\_XFRSTAT\_CLR register.

## Using DMA

1. Write the RSI\_ARG register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the RSI\_ARG register.
2. Write the RSI\_CMD register with the SELECT/DESELECT\_CARD command, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1b.
3. Wait for the RESPEND indication within the RSI\_XFRSTAT register, and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register.
4. Ensure that the device is not busy and no errors occurred by verifying the response contained in RSI\_RESP0.
5. Configure the DMA channel assigned to the RSI controller. Write DMAx\_ADDRSTART with the address of the first byte of data to be written to the card. The DMAx\_XCNT register should be set to the overall number of 32-bit words to be written; for example, write 1024 to transfer 4096 bytes. The DMAx\_XMOD register should be set to 4. The DMAx\_CFG register should be set for DMA enable and a word size of 32-bits.
6. Once the DMA channel has been configured and enabled, write the number of bytes to be transferred to the RSI\_DATA\_LEN register. For example, write 4096 to write eight blocks of 512 bytes.
7. Write the appropriate timeout value for a write operation to the RSI\_DATA\_TMR register.

8. Write the destination start address to the `RSI_ARG` register. The supplied address must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or sector addressable MMC card.
9. Write the `WRITE_MULTIPLE_BLOCK` command to the `RSI_CMD`, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
10. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register, and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
11. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN` and `DMAEN` should also be set to enable the data path state machine and to allow the DMA controller to access the transmit FIFO. All other fields of the `RSI_DATA_CTL` register should be zero.
12. Poll for the `DATEND` flag or alternatively poll for each instance of the `DATBLKEND` flag that will be set on successful completion of each block transfer. For a 4096 byte transfer, `DATBLKEND` will be set eight times and should be cleared after it is detected via the `RSI_XFRSTAT_CLR` register.
13. Write the `RSI_CMD` register with the `STOP_TRANSMISSION` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
14. Clear the `DATEND` and `RESPEND` flag via the `RSI_XFRSTAT_CLR` register. Also clear the `IRQDONE` bit of the `DMAx_STAT` register, if applicable.

## Multiple Block Read

Block read operations typically consist of 512 bytes of data per block. If the card is found to support other block lengths or the default block length as specified in the CID register is not 512, the block length of the RSI must be configured accordingly. The block length of the card and the block length of the RSI must be configured for the same block size at all times. The block length of the RSI is configured via the `RSI_BLKSZ` field of the `RSI_DATA_CTL` register.

## Using the Core

1. Write the `RSI_ARG` register with the card's RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.
3. Wait for the `RESPEND` indication within the `RSI_XFRSTAT` register, and clear the status bit once detected via the `RSI_XFRSTAT_CLR` register.
4. Ensure that the device is not busy and there are no errors occurred by verifying the response contained in `RSI_RESP0`.

5. Write the number of bytes to be transferred to the `RSI_DATA_LEN` register. This will be 4096 for a transfer of eight 512 byte blocks.
6. Write the appropriate timeout value for a read operation to the `RSI_DATA_TMR` register.
7. Write the destination start address to the `RSI_ARG` register. The supplied address must be aligned to a 512-byte boundary if misaligned accesses are not enabled and the card is not a high-capacity SD card or a sector-addressable MMC card.
8. Enable the data path state machine by writing to the `RSI_DATA_CTL` register with `RSI_BLKSZ` set to 512 for a 512-byte block. `DATEN` and `DATDIR` should also be set to enable the data path state machine and indicate the transfer direction is from card to controller. All other fields of the `RSI_DATA_CTL` register should be zero.
9. Write the `READ_MULTIPLE_BLOCK` command to the `RSI_CMD` register, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
10. In order to meet some timing restrictions related to block read operations, it is advisable to not wait for the `RESPEND` indication within the `RSI_XFRSTAT` register but instead move immediately on the next step. This is due to the card being able to send data before a response can completed on the `RSI_CMD` signal, moving immediately onto step 11 will ensure a receive FIFO overflow does not occur.
11. Poll the `RXFIFORDY` bit or the `RXFIFOZERO` bit of `RSI_XFRSTAT` indicating the receive FIFO has data available, or the receive FIFO is empty. As long as the receive FIFO is not empty, read data from the `RSI_FIFO` register until 512 bytes have been read.
12. Once the block has been read, wait for the `DATBLKEND` flag to indicate that the data was received correctly and passed the CRC check.
13. Clear the `DATBLKEND` flag via `RSI_XFRSTAT_CLR`.
14. Repeat steps 11 to 13 until the required number of blocks have been read or until the `DATEND` flag has been set.
15. Write the `RSI_CMD` register with the `STOP_TRANSMISSION` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1`.
16. Clear the `DATEND` and `RESPEND` flags via the `RSI_XFRSTAT_CLR` register.

## Using DMA

1. Write the `RSI_ARG` register with the cards RCA. The 16-bit RCA should be written to the upper 16-bits of the `RSI_ARG` register.
2. Write the `RSI_CMD` register with the `SELECT/DESELECT_CARD` command, configuring the command path state machine to expect a short response by setting `RSP` and clearing `LRSP`. The response type is `R1b`.

3. Wait for the RESPEND indication within the RSI\_XFRSTAT register, and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register.
4. Ensure that the device is not busy and that no errors occurred by verifying the response contained in RSI\_RESP0.
5. Configure the DMA channel assigned to the RSI controller. Write DMAx\_ADDRSTART with the address of the first byte of where the received data is to be stored. The DMAx\_XCNT register should be set to the number of 32-bit words to be read, which would be 1024 for a 4096 byte read transfer. The DMAx\_XMOD register should be set to 4. The DMAx\_CFG register should be set for DMA enable (a word size of 32-bits and direction set to memory write).
6. Write the number of bytes to be transferred to the RSI\_DATA\_LEN register. This will be 4096 for eight blocks of 512 bytes.
7. Write the appropriate timeout value for a read operation to the RSI\_DATA\_TMR register.
8. Write the source start address to the RSI\_ARG register. The supplied address must be aligned to a 512-byte boundary if mis-aligned accesses are not enabled and the card is not a high-capacity SD card or a sector-addressable MMC card.
9. Enable the data path state machine by writing to the RSI\_DATA\_CTL register with RSI\_BLKSZ set to 512 for a 512-byte block. DATEN, DATDIR, and DMAEN should also be set to enable the data path state machine. Set the transfer direction from card to controller and allow the DMA controller access to the receive FIFO. All other fields of the RSI\_DATA\_CTL register should be zero.
10. Write the READ\_MULTIPLE\_BLOCK command to the RSI\_CMD register, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1.
11. Unlike core accesses, it is safe to poll on RESPEND indication within the RSI\_XFRSTAT register and clear the status bit once detected via the RSI\_XFRSTAT\_CLR register. The DMA controller, enabled in step 5 will ensure any data sent to the receive FIFO prior to the RESPEND flag being set is received correctly.
12. Poll for the DATEND flag or alternatively poll for each instance of the DATBLKEND flag that will be set on successful completion of each block transfer. For a 4096-byte transfer, DATBLKEND will be set eight times and should be cleared after it is detected via the RSI\_XFRSTAT\_CLR register.
13. Write the RSI\_CMD register with the STOP\_TRANSMISSION command, configuring the command path state machine to expect a short response by setting RSP and clearing LRSP. The response type is R1.
14. Clear the DATEND and RESPEND flags via the RSI\_XFRSTAT\_CLR register. Also clear the IRQDONE bit of the DMAx\_STAT register, if applicable.

## RSI Programming Concepts

This section provides details about some special programming considerations to effectively use the RSI module.

## Disabling CRC check

According to specifications of memory cards such as eMMC, SDIO, and others, the cards do not send CRC for few selected commands. RSI in earlier Blackfin processors takes the stuffed bits coming from the card as CRC and gives a CRC check failure interrupt. To overcome this, a bit is added in the RSI\_CMD register where software needs to program the bit `CRCDIS` whether to disable CRC check or not.

## Data End Interrupt

In older RSI versions, DataEnd Interrupt is raised when the Data Counter reaches 0. This has been modified in the new version of RSI. DataEnd interrupt is delayed in data write operation if the card indicates a busy condition by pulling the DAT0 line low. This interrupt will be raised once the DAT0 line is released to high.

## Miscellaneous Programming Guidelines

- `PORTx_FER` bit for `RSI_CLK` should be set before `RSI_DATA3`.
- `RSI_CMD` and `RSI_DATA0` will always be used together. Therefor only one pull-up enable is used for these 2 lines.
- Write 1 to clear the SD card detect interrupt status bit, i.e bit 4 of the `RSI_STAT0` register after SDMMC reset is issued.
- There is a 2.5 system clock cycle latency involved with a write to the RSI registers. Any action that needs to be taken after the register write should be timed accordingly. This has to be specially kept in mind in scenarios where the system clock on which RSI operates is very slow compared to the core domain clock.

## RSI Programming Examples

This section is TBD

## RSI Register Descriptions

RSI (RSI) contains the following registers.

**Table 24-17:** RSI Module Registers

| Name              | Description                               |
|-------------------|-------------------------------------------|
| RSI_CTL           | Control Register                          |
| RSI_ARG           | Argument Register                         |
| RSI_CMD           | Command Register                          |
| RSI_RESP_CMD      | Response Command Register                 |
| RSI_RESP0         | Response 0 Register                       |
| RSI_RESP1         | Response 1 Register                       |
| RSI_RESP2         | Response 2 Register                       |
| RSI_RESP3         | Response 3 Register                       |
| RSI_DATA_TMR      | Data Timer Register                       |
| RSI_DATA_LEN      | Data Length Register                      |
| RSI_DATA_CTL      | Data Control Register                     |
| RSI_DATA_CNT      | Data Count Register                       |
| RSI_XFRSTAT       | Status Register                           |
| RSI_XFRSTAT_CLR   | Status Clear Register                     |
| RSI_XFR_IMSK0     | Interrupt 0 Mask Register                 |
| RSI_XFR_IMSK1     | Interrupt 1 Mask Register                 |
| RSI_FIFO_CNT      | FIFO Counter Register                     |
| RSI_CEATA         | This register contains bit to dis CCS gen |
| RSI_BOOT_TCNTR    | Boot Timing Counter Register              |
| RSI_BACK_TOUT     | Boot Acknowledge Timeout Register         |
| RSI_SLP_WKUP_TOUT | Sleep Wakeup Timeout Register             |
| RSI_BLKSZ         | Block Size Register                       |

Table 24-17: RSI Module Registers (Continued)

| Name        | Description                        |
|-------------|------------------------------------|
| RSI_FIFO    | Data FIFO Register                 |
| RSI_STAT0   | Exception Status Register          |
| RSI_IMSK0   | Exception Mask Register            |
| RSI_CFG     | Configuration Register             |
| RSI_RD_WAIT | Read Wait Enable Register          |
| RSI_PID0    | Peripheral Identification Register |
| RSI_PID1    | Peripheral Identification Register |
| RSI_PID2    | Peripheral Identification Register |
| RSI_PID3    | Peripheral Identification Register |

Control Register

RSI\_CTL: Control Register - R/W

Reset = 0x0000

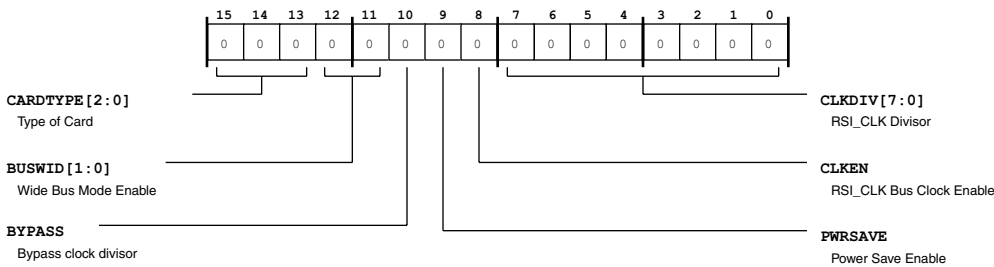


Figure 24-6: RSI\_CTL Register Diagram



**Table 24-18:** RSI\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                     |
|---------|----------|---------------------------------------------|
| 15:13   | CARDTYPE | Type of Card.                               |
|         |          | 0 SDIO                                      |
|         |          | 1 eMMC                                      |
|         |          | 1xx Reserved                                |
|         |          | 2 SD Card                                   |
|         |          | 3 CE-ATA                                    |
| 12:11   | BUSWID   | Wide Bus Mode Enable.                       |
|         |          | 0 Standard bus mode (only RSI_D0 used)      |
|         |          | 1 Wide bus mode (4bit bus)                  |
|         |          | 2 Byte bus mode (8bit bus)                  |
|         |          | 3 Reserved                                  |
| 10      | BYPASS   | Bypass clock divisor.                       |
|         |          | 0 Disable bypass                            |
|         |          | 1 Enable bypass (RSI_CLK driven from SCLK0) |
| 9       | PWRSERVE | Power Save Enable.                          |
|         |          | 0 Always Enabled                            |
|         |          | 1 Clock enabled when bus is active          |
| 8       | CLKEN    | RSI_CLK Bus Clock Enable.                   |
|         |          | 0 Disabled                                  |
|         |          | 1 Enabled                                   |
| 7:0     | CLKDIV   | RSI_CLK Divisor.                            |

Argument Register

RSI\_ARG: Argument Register - R/W

Reset = 0x0000 0000

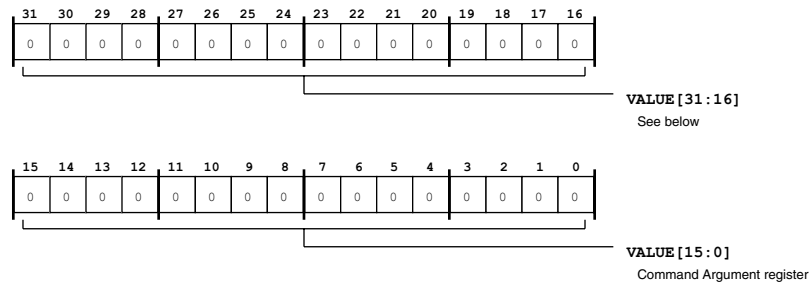


Figure 24-7: RSI\_ARG Register Diagram

Table 24-19: RSI\_ARG Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Command Argument register. |

Command Register

RSI\_CMD: Command Register - R/W

Reset = 0x0000

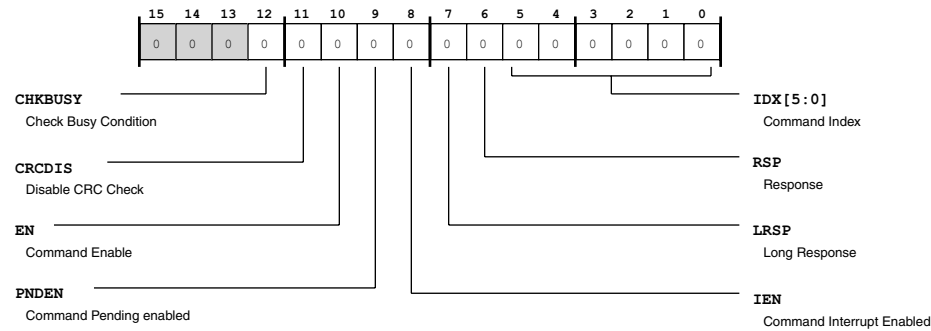


Figure 24-8: RSI\_CMD Register Diagram

**Table 24-20:** RSI\_CMD Register Fields

| Bit No. | Bit Name | Description/Enumeration    |                                                                                                              |
|---------|----------|----------------------------|--------------------------------------------------------------------------------------------------------------|
| 12      | CHKBUSY  | Check Busy Condition.      |                                                                                                              |
|         |          | 0                          | Don't check for Busy state on the DATA0 pin                                                                  |
|         |          | 1                          | Check for Busy state on the DATA0 pin                                                                        |
| 11      | CRCDIS   | Disable CRC Check.         |                                                                                                              |
|         |          | 0                          | CRC Check is not disabled                                                                                    |
|         |          | 1                          | CRC Check is disabled                                                                                        |
| 10      | EN       | Command Enable.            |                                                                                                              |
|         |          | 0                          | Disable command state machine                                                                                |
|         |          | 1                          | Enable command state machine                                                                                 |
| 9       | PNDEN    | Command Pending enabled.   |                                                                                                              |
|         |          | 1                          | Command state machine waits for CmdPend before it starts sending a command. Used only during MMC Stream Mode |
| 8       | IEN      | Command Interrupt Enabled. |                                                                                                              |
|         |          | 1                          | Command state machine disables command timer and waits for interrupt request                                 |
| 7       | LRSP     | Long Response.             |                                                                                                              |
|         |          | 0                          | Command state machine receives a short response                                                              |
|         |          | 1                          | Command state machine receives a 136-bit long response                                                       |

Table 24-20: RSI\_CMD Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                      |
|---------|----------|----------------------------------------------|
| 6       | RSP      | Response.                                    |
|         |          | 0 No response, expect CmdSent flag           |
|         |          | 1 Command state machine waits for a response |
| 5:0     | IDX      | Command Index.                               |

Response Command Register

RSI\_RESP\_CMD: Response Command Register - R/NW

Reset = 0x0000

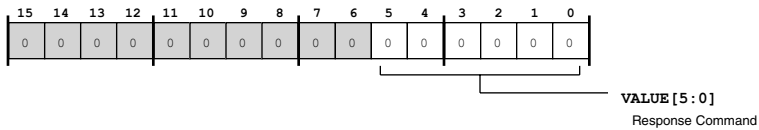


Figure 24-9: RSI\_RESP\_CMD Register Diagram

Table 24-21: RSI\_RESP\_CMD Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 5:0     | VALUE    | Response Command.       |

Response 0 Register

RSI\_RESP0: Response 0 Register - R/NW

Reset = 0x0000 0000

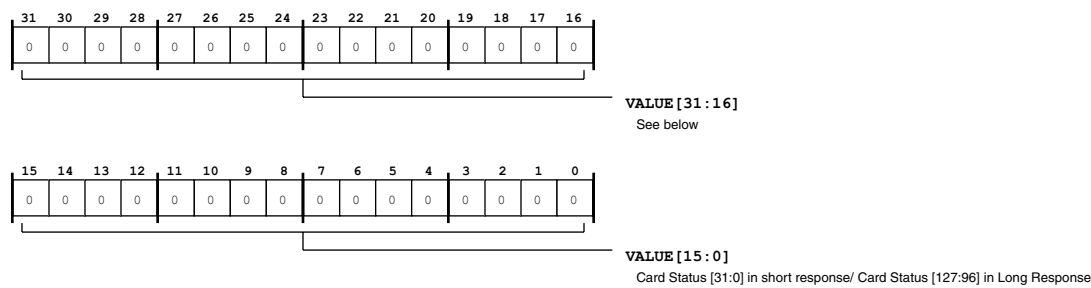


Figure 24-10: RSI\_RESP0 Register Diagram

Table 24-22: RSI\_RESP0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                      |
|---------|----------|------------------------------------------------------------------------------|
| 31:0    | VALUE    | Card Status [31:0] in short response/ Card Status [127:96] in Long Response. |

Response 1 Register

RSI\_RESP1: Response 1 Register - R/NW

Reset = 0x0000 0000

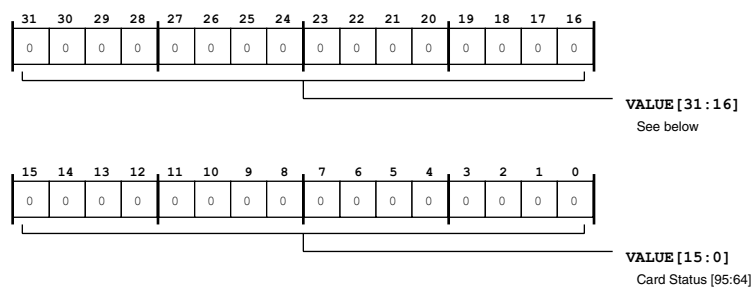


Figure 24-11: RSI\_RESP1 Register Diagram

Table 24-23: RSI\_RESP1 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Card Status [95:64].    |

## Response 2 Register

RSI\_RESP2: Response 2 Register - R/NW

Reset = 0x0000 0000

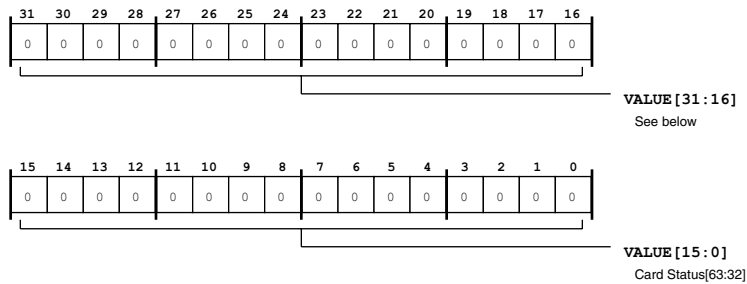


Figure 24-12: RSI\_RESP2 Register Diagram

Table 24-24: RSI\_RESP2 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Card Status[63:32].     |

## Response 3 Register

The Most Significant Bit of the card status is received first. The Resp3 register Most Significant Bit is always zero.

RSI\_RESP3: Response 3 Register - R/NW

Reset = 0x0000 0000

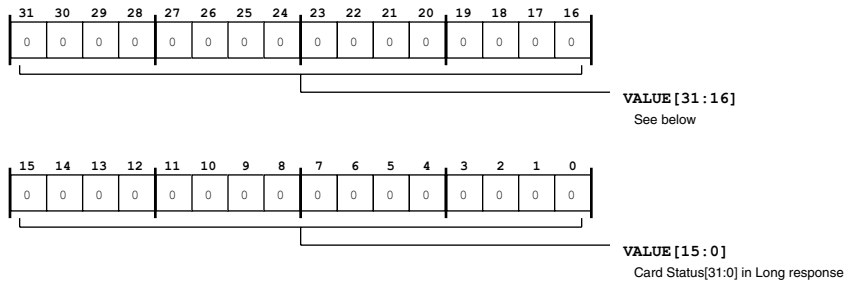


Figure 24-13: RSI\_RESP3 Register Diagram

**Table 24-25:** RSI\_RESP3 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                           |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Card Status[31:0] in Long response.<br>The Most Significant Bit of the card status is received first. The Response3 register Most Significant Bit is always zero. |

## Data Timer Register

RSI\_DATA\_TMR: Data Timer Register - R/W

Reset = 0x0000 0000

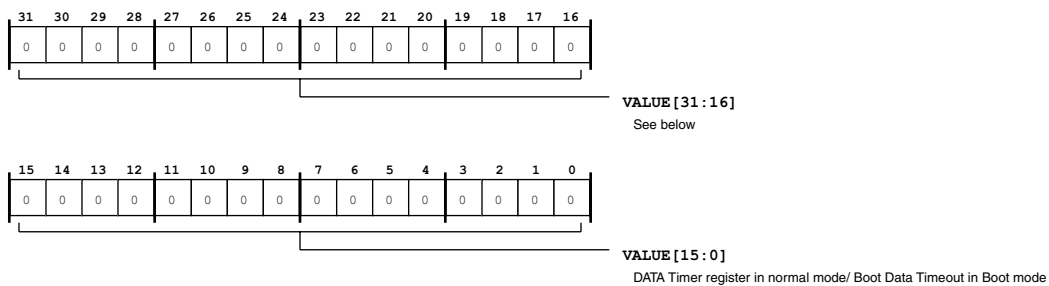


**Figure 24-14:** RSI\_DATA\_TMR Register Diagram

**Table 24-26:** RSI\_DATA\_TMR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                             |
|---------|----------|---------------------------------------------------------------------|
| 31:0    | VALUE    | DATA Timer register in normal mode/ Boot Data Timeout in Boot mode. |

## Data Length Register

RSI\_DATA\_LEN: Data Length Register - R/W

Reset = 0x0000

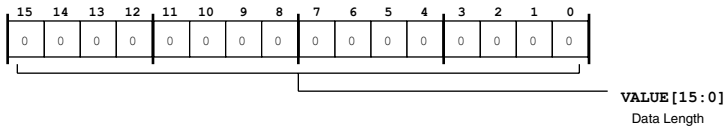


Figure 24-15: RSI\_DATA\_LEN Register Diagram

Table 24-27: RSI\_DATA\_LEN Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Data Length.            |

Data Control Register

RSI\_DATA\_CTL: Data Control Register - R/W

Reset = 0x0000

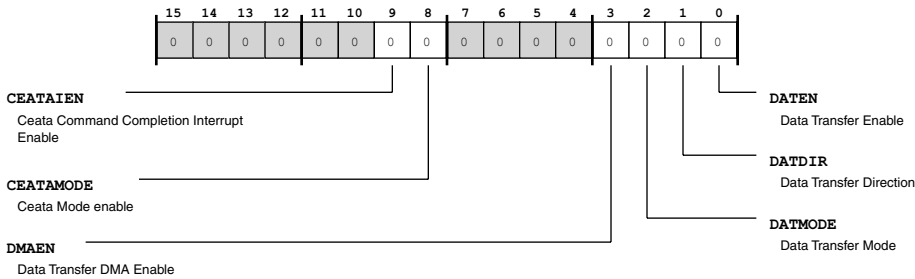


Figure 24-16: RSI\_DATA\_CTL Register Diagram

Table 24-28: RSI\_DATA\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                    |
|---------|----------|--------------------------------------------|
| 9       | CEATAIEN | Ceata Command Completion Interrupt Enable. |
|         |          | 0 No CCS from device                       |
|         |          | 1 CCS from device                          |



**Table 24-28:** RSI\_DATA\_CTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration   |                         |
|---------|-----------|---------------------------|-------------------------|
| 8       | CEATAMODE | Ceata Mode enable.        |                         |
|         |           | 0                         | No CE-ATA operation     |
|         |           | 1                         | CE-ATA operation        |
| 3       | DMAEN     | Data Transfer DMA Enable. |                         |
|         |           | 0                         | DMA Disabled            |
|         |           | 1                         | DMA Enabled             |
| 2       | DATMODE   | Data Transfer Mode.       |                         |
|         |           | 0                         | Block data transfer     |
|         |           | 1                         | Stream data transfer    |
| 1       | DATDIR    | Data Transfer Direction.  |                         |
|         |           | 0                         | From controller to card |
|         |           | 1                         | From card to controller |
| 0       | DATEN     | Data Transfer Enable.     |                         |
|         |           | 0                         | Data transfer disabled  |
|         |           | 1                         | Data transfer enabled   |

Data Count Register

RSI\_DATA\_CNT: Data Count Register - R/NW

Reset = 0x0000

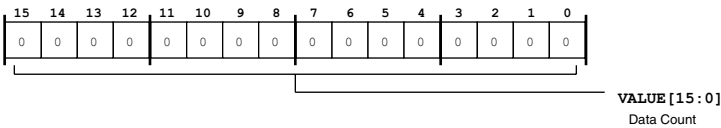


**Figure 24-17:** RSI\_DATA\_CNT Register Diagram

Table 24-29: RSI\_DATA\_CNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Data Count.             |

Status Register

RSI\_XFRSTAT: Status Register - R/NW

Reset = 0x0000 0000

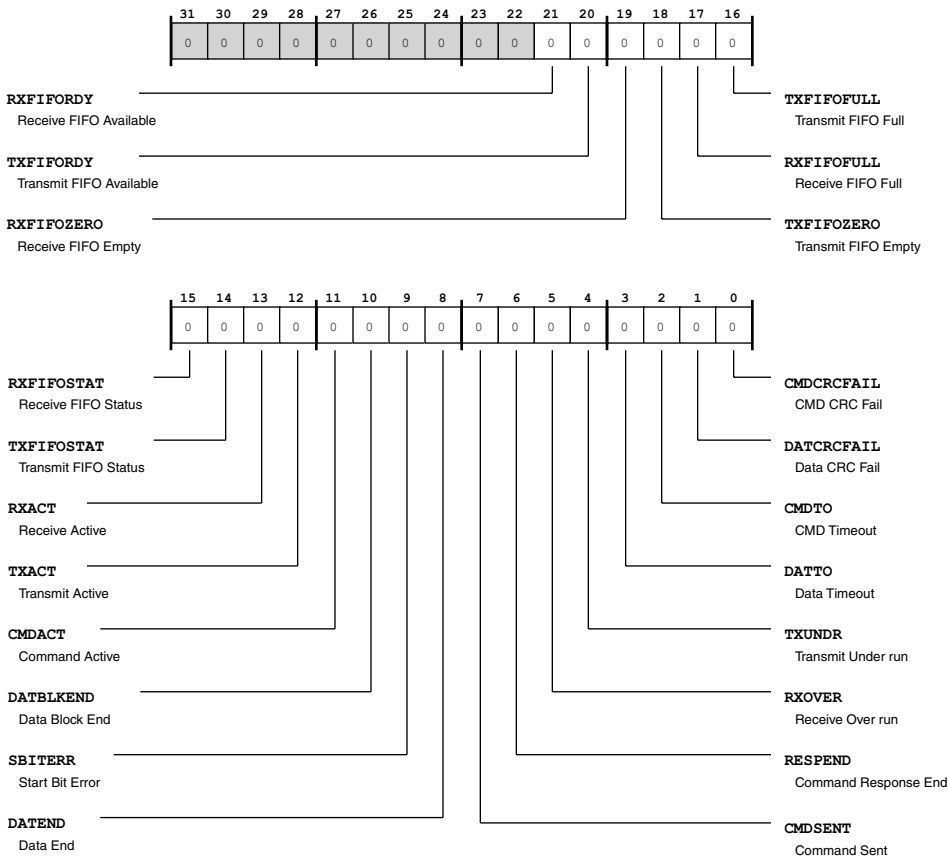


Figure 24-18: RSI\_XFRSTAT Register Diagram

Table 24-30: RSI\_XFRSTAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration |
|---------|-----------|-------------------------|
| 21      | RXFIFORDY | Receive FIFO Available. |

**Table 24-30:** RSI\_XFRSTAT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration  |
|---------|------------|--------------------------|
| 20      | TXFIFORDY  | Transmit FIFO Available. |
| 19      | RXFIFOZERO | Receive FIFO Empty.      |
| 18      | TXFIFOZERO | Transmit FIFO Empty.     |
| 17      | RXFIFOFULL | Receive FIFO Full.       |
| 16      | TXFIFOFULL | Transmit FIFO Full.      |
| 15      | RXFIFOSTAT | Receive FIFO Status.     |
| 14      | TXFIFOSTAT | Transmit FIFO Status.    |
| 13      | RXACT      | Receive Active.          |
| 12      | TXACT      | Transmit Active.         |
| 11      | CMDACT     | Command Active.          |
| 10      | DATBLKEND  | Data Block End.          |
| 9       | SBITERR    | Start Bit Error.         |
| 8       | DATEND     | Data End.                |
| 7       | CMDSENT    | Command Sent.            |
| 6       | RESPEND    | Command Response End.    |
| 5       | RXOVER     | Receive Over run.        |
| 4       | TXUNDR     | Transmit Under run.      |
| 3       | DATTO      | Data Timeout.            |
| 2       | CMDTO      | CMD Timeout.             |
| 1       | DATCRCFAIL | Data CRC Fail.           |
| 0       | CMDCRCFAIL | CMD CRC Fail.            |

Status Clear Register

RSI\_XFRSTAT\_CLR: Status Clear Register - R/W

Reset = 0x0000

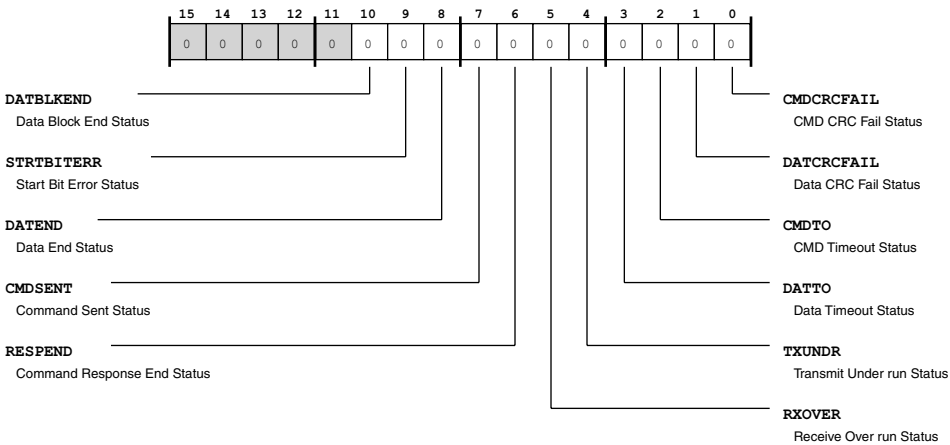


Figure 24-19: RSI\_XFRSTAT\_CLR Register Diagram

Table 24-31: RSI\_XFRSTAT\_CLR Register Fields

| Bit No. | Bit Name   | Description/Enumeration      |
|---------|------------|------------------------------|
| 10      | DATBLKEND  | Data Block End Status.       |
| 9       | STRTBITERR | Start Bit Error Status.      |
| 8       | DATEND     | Data End Status.             |
| 7       | CMDSSENT   | Command Sent Status.         |
| 6       | RESPEND    | Command Response End Status. |
| 5       | RXOVER     | Receive Over run Status.     |
| 4       | TXUNDR     | Transmit Under run Status.   |
| 3       | DATTO      | Data Timeout Status.         |
| 2       | CMDTO      | CMD Timeout Status.          |
| 1       | DATCRCFAIL | Data CRC Fail Status.        |
| 0       | CMDCRCFAIL | CMD CRC Fail Status.         |

# Interrupt 0 Mask Register

RSI\_XFR\_IMSK0: Interrupt 0 Mask Register - R/W

Reset = 0x0000 0000

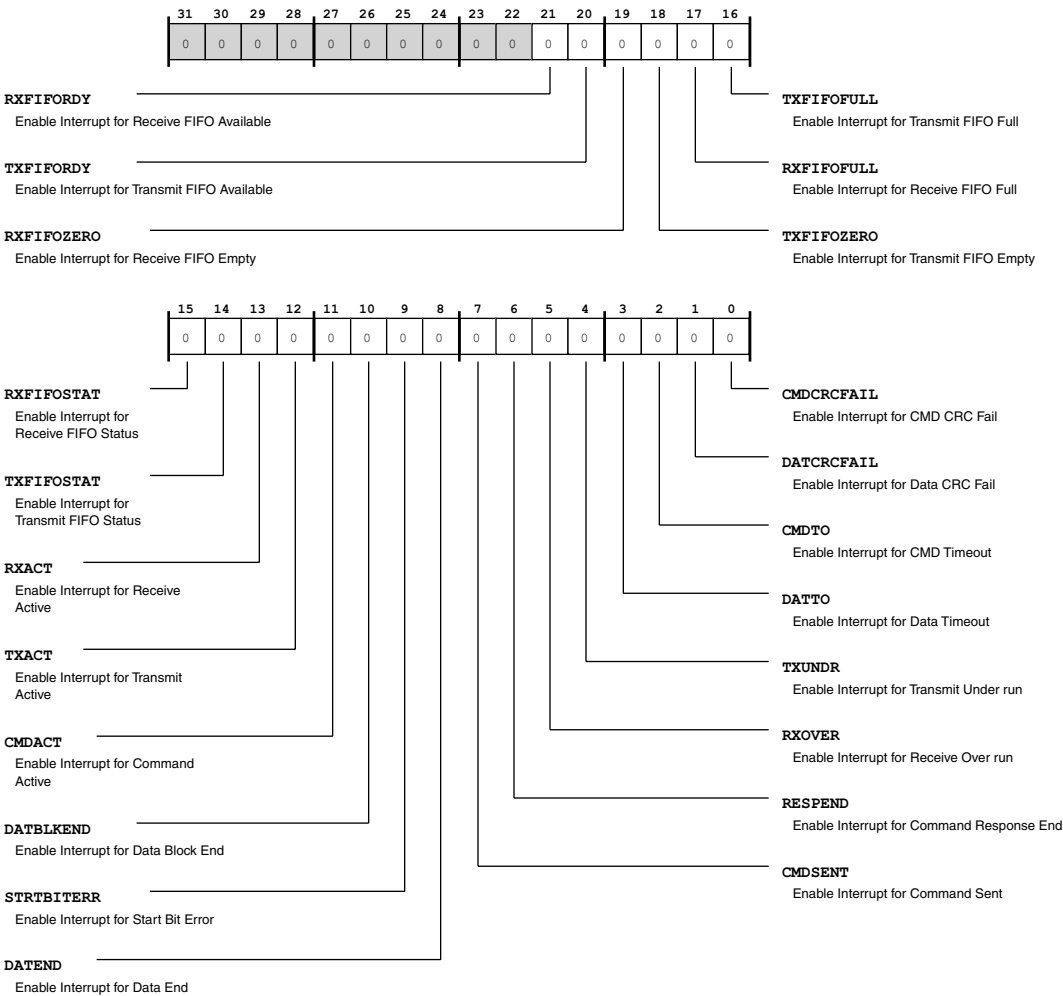


Figure 24-20: RSI\_XFR\_IMSK0 Register Diagram

Table 24-32: RSI\_XFR\_IMSK0 Register Fields

| Bit No. | Bit Name   | Description/Enumeration                       |
|---------|------------|-----------------------------------------------|
| 21      | RXFIFORDY  | Enable Interrupt for Receive FIFO Available.  |
| 20      | TXFIFORDY  | Enable Interrupt for Transmit FIFO Available. |
| 19      | RXFIFOZERO | Enable Interrupt for Receive FIFO Empty.      |

**Table 24-32:** RSI\_XFR\_IMSK0 Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                    |
|---------|------------|--------------------------------------------|
| 18      | TXFIFOZERO | Enable Interrupt for Transmit FIFO Empty.  |
| 17      | RXFIFOFULL | Enable Interrupt for Receive FIFO Full.    |
| 16      | TXFIFOFULL | Enable Interrupt for Transmit FIFO Full.   |
| 15      | RXFIFOSTAT | Enable Interrupt for Receive FIFO Status.  |
| 14      | TXFIFOSTAT | Enable Interrupt for Transmit FIFO Status. |
| 13      | RXACT      | Enable Interrupt for Receive Active.       |
| 12      | TXACT      | Enable Interrupt for Transmit Active.      |
| 11      | CMDACT     | Enable Interrupt for Command Active.       |
| 10      | DATBLKEND  | Enable Interrupt for Data Block End.       |
| 9       | STRTBITERR | Enable Interrupt for Start Bit Error.      |
| 8       | DATEND     | Enable Interrupt for Data End.             |
| 7       | CMDSENT    | Enable Interrupt for Command Sent.         |
| 6       | RESPEND    | Enable Interrupt for Command Response End. |
| 5       | RXOVER     | Enable Interrupt for Receive Over run.     |
| 4       | TXUNDR     | Enable Interrupt for Transmit Under run.   |
| 3       | DATTO      | Enable Interrupt for Data Timeout.         |
| 2       | CMDTO      | Enable Interrupt for CMD Timeout.          |
| 1       | DATCRCFAIL | Enable Interrupt for Data CRC Fail.        |
| 0       | CMDCRCFAIL | Enable Interrupt for CMD CRC Fail.         |

## Interrupt 1 Mask Register

### RSI\_XFR\_IMSK1: Interrupt 1 Mask Register - R/W

Reset = 0x0000 0000

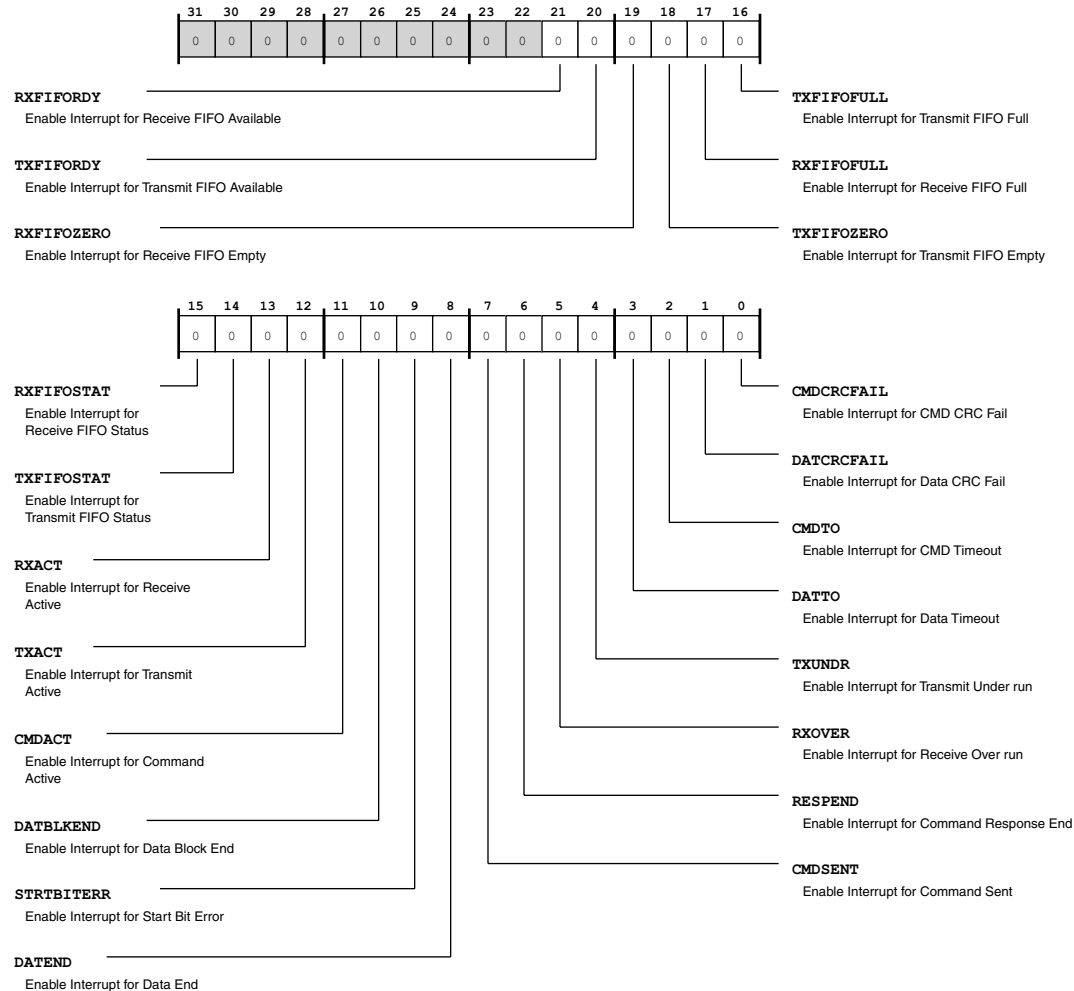


**Figure 24-21:** RSI\_XFR\_IMSK1 Register Diagram

**Table 24-33:** RSI\_XFR\_IMSK1 Register Fields

| Bit No. | Bit Name   | Description/Enumeration                       |
|---------|------------|-----------------------------------------------|
| 21      | RXFIFORDY  | Enable Interrupt for Receive FIFO Available.  |
| 20      | TXFIFORDY  | Enable Interrupt for Transmit FIFO Available. |
| 19      | RXFIFOZERO | Enable Interrupt for Receive FIFO Empty.      |
| 18      | TXFIFOZERO | Enable Interrupt for Transmit FIFO Empty.     |

**Table 24-33:** RSI\_XFR\_IMSK1 Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                    |
|---------|------------|--------------------------------------------|
| 17      | RXFIFOFULL | Enable Interrupt for Receive FIFO Full.    |
| 16      | TXFIFOFULL | Enable Interrupt for Transmit FIFO Full.   |
| 15      | RXFIFOSTAT | Enable Interrupt for Receive FIFO Status.  |
| 14      | TXFIFOSTAT | Enable Interrupt for Transmit FIFO Status. |
| 13      | RXACT      | Enable Interrupt for Receive Active.       |
| 12      | TXACT      | Enable Interrupt for Transmit Active.      |
| 11      | CMDACT     | Enable Interrupt for Command Active.       |
| 10      | DATBLKEND  | Enable Interrupt for Data Block End.       |
| 9       | STRTBITERR | Enable Interrupt for Start Bit Error.      |
| 8       | DATEND     | Enable Interrupt for Data End.             |
| 7       | CMDSENT    | Enable Interrupt for Command Sent.         |
| 6       | RESPEND    | Enable Interrupt for Command Response End. |
| 5       | RXOVER     | Enable Interrupt for Receive Over run.     |
| 4       | TXUNDR     | Enable Interrupt for Transmit Under run.   |
| 3       | DATTO      | Enable Interrupt for Data Timeout.         |
| 2       | CMDTO      | Enable Interrupt for CMD Timeout.          |
| 1       | DATCRCFAIL | Enable Interrupt for Data CRC Fail.        |
| 0       | CMDCRCFAIL | Enable Interrupt for CMD CRC Fail.         |

## FIFO Counter Register



RSI\_FIFO\_CNT: FIFO Counter Register - R/W

Reset = 0x0000

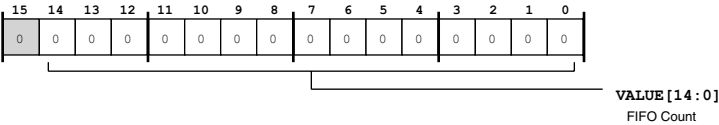


Figure 24-22: RSI\_FIFO\_CNT Register Diagram

Table 24-34: RSI\_FIFO\_CNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 14:0    | VALUE    | FIFO Count.             |

This register contains bit to dis CCS gen

RSI\_CEATA: This register contains bit to dis CCS gen - R/W

Reset = 0x0000 0000

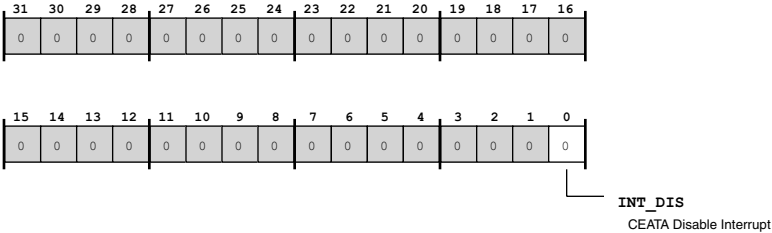


Figure 24-23: RSI\_CEATA Register Diagram

Table 24-35: RSI\_CEATA Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 0       | INT_DIS  | CEATA Disable Interrupt. |

Boot Timing Counter Register

RSI\_BOOT\_TCNTR: Boot Timing Counter Register - R/W

Reset = 0x384a

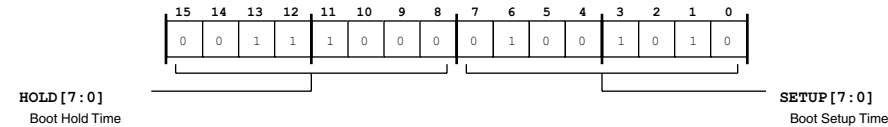


Figure 24-24: RSI\_BOOT\_TCNTR Register Diagram

Table 24-36: RSI\_BOOT\_TCNTR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:8    | HOLD     | Boot Hold Time.         |
| 7:0     | SETUP    | Boot Setup Time.        |

Boot Acknowledge Timeout Register

RSI\_BACK\_TOUT: Boot Acknowledge Timeout Register - R/W

Reset = 0x0000 0000

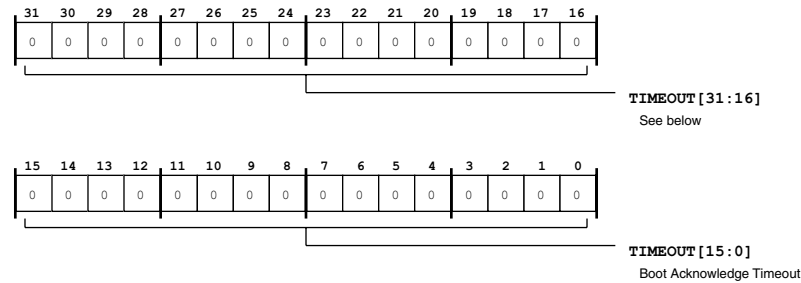


Figure 24-25: RSI\_BACK\_TOUT Register Diagram

Table 24-37: RSI\_BACK\_TOUT Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | TIMEOUT  | Boot Acknowledge Timeout. |

# Sleep Wakeup Timeout Register

RSI\_SLP\_WKUP\_TOUT: Sleep Wakeup Timeout Register - R/W

Reset = 0x0000 0000

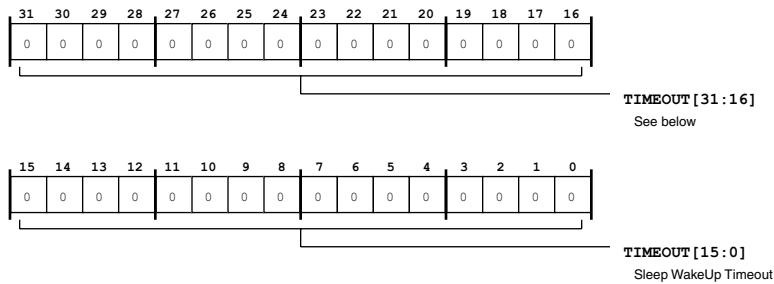


Figure 24-26: RSI\_SLP\_WKUP\_TOUT Register Diagram

Table 24-38: RSI\_SLP\_WKUP\_TOUT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | TIMEOUT  | Sleep WakeUp Timeout.   |

# Block Size Register

RSI\_BLKSZ: Block Size Register - R/W

Reset = 0x0000

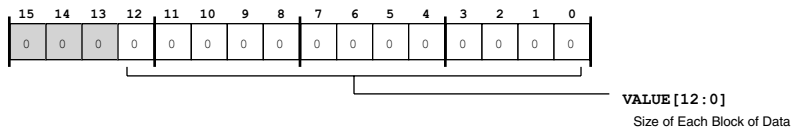


Figure 24-27: RSI\_BLKSZ Register Diagram

Table 24-39: RSI\_BLKSZ Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 12:0    | VALUE    | Size of Each Block of Data. |

Data FIFO Register

RSI\_FIFO: Data FIFO Register - R/W

Reset = 0x0000 0000

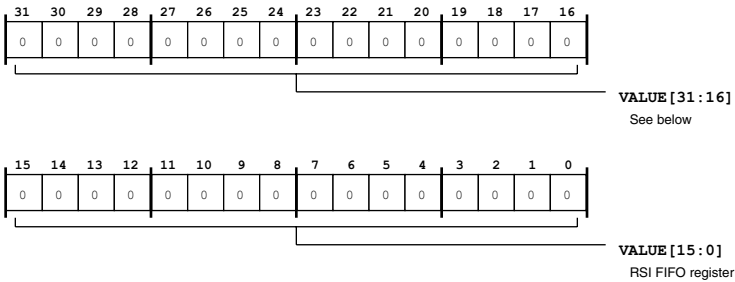


Figure 24-28: RSI\_FIFO Register Diagram

Table 24-40: RSI\_FIFO Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | RSI FIFO register.      |

Exception Status Register

**RSI\_STAT0: Exception Status Register - R/W**

Reset = 0x0000 0000

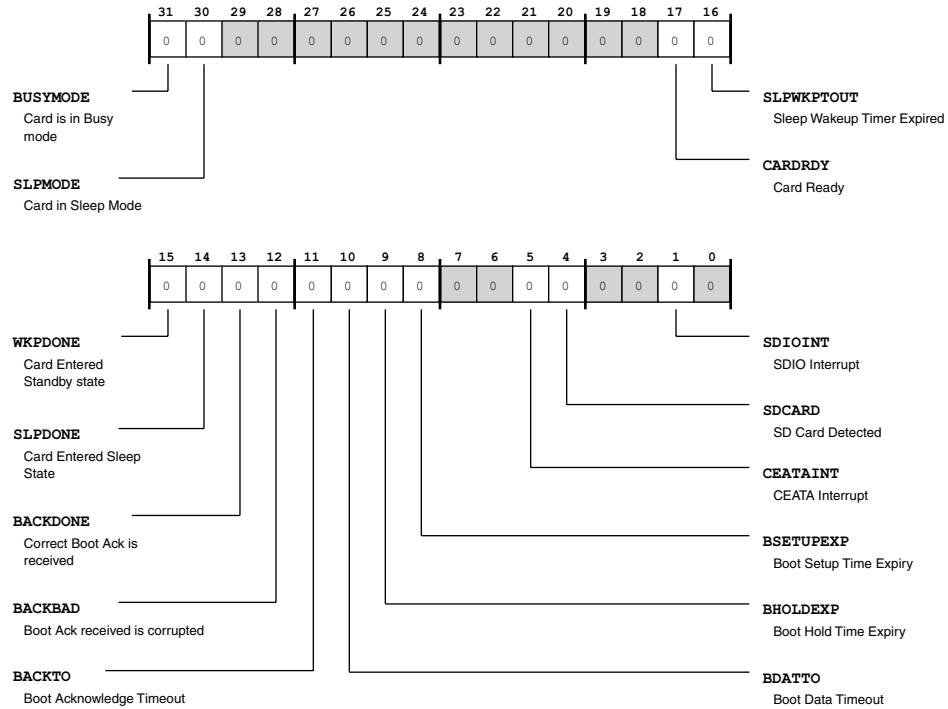


**Figure 24-29:** RSI\_STAT0 Register Diagram

**Table 24-41:** RSI\_STAT0 Register Fields

| Bit No. | Bit Name   | Description/Enumeration          |
|---------|------------|----------------------------------|
| 31      | BUSYMODE   | Card is in Busy mode.            |
|         |            | 1 Card is in Busy Mode           |
| 30      | SLPMODE    | Card in Sleep Mode.              |
|         |            | 1 Card is in Sleep Mode          |
| 17      | CARDRDY    | Card Ready.                      |
|         |            | 1 Card has come out of busy mode |
| 16      | SLPWKPTOUT | Sleep Wakeup Timer Expired.      |
|         |            | 1 Sleep Wakeup Timer expired     |

**Table 24-41:** RSI\_STAT0 Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration         |                                                          |
|---------|-----------|---------------------------------|----------------------------------------------------------|
| 15      | WKPDONE   | Card Entered Standby state.     |                                                          |
|         |           | 1                               | Card successfully entered Standby State from Sleep State |
| 14      | SLPDONE   | Card Entered Sleep State.       |                                                          |
|         |           | 1                               | Card successfully entered Sleep State from Standby State |
| 13      | BACKDONE  | Correct Boot Ack is received.   |                                                          |
|         |           | 1                               | Correct Boot Ack received                                |
| 12      | BACKBAD   | Boot Ack received is corrupted. |                                                          |
|         |           | 1                               | Boot Ack received is corrupted                           |
| 11      | BACKTO    | Boot Acknowledge Timeout.       |                                                          |
|         |           | 1                               | Timed out                                                |
| 10      | BDATTO    | Boot Data Timeout.              |                                                          |
|         |           | 1                               | Timed out                                                |
| 9       | BHOLDEXP  | Boot Hold Time Expiry.          |                                                          |
|         |           | 1                               | Counter Expired                                          |
| 8       | BSETUPEXP | Boot Setup Time Expiry.         |                                                          |
|         |           | 1                               | Counter Expired                                          |
| 5       | CEATAINT  | CEATA Interrupt.                |                                                          |
|         |           | 0                               | No Interrupt Detected                                    |
|         |           | 1                               | Interrupt Detected                                       |

**Table 24-41:** RSI\_STAT0 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 4       | SDCARD   | SD Card Detected.       |
|         |          | 0 No SD Card Detected   |
|         |          | 1 SD Card Detected      |
| 1       | SDIOINT  | SDIO Interrupt.         |
|         |          | 0 No interrupt detected |
|         |          | 1 Interrupt detected    |

## Exception Mask Register

### RSI\_IMSK0: Exception Mask Register - R/W

Reset = 0x0000 0000

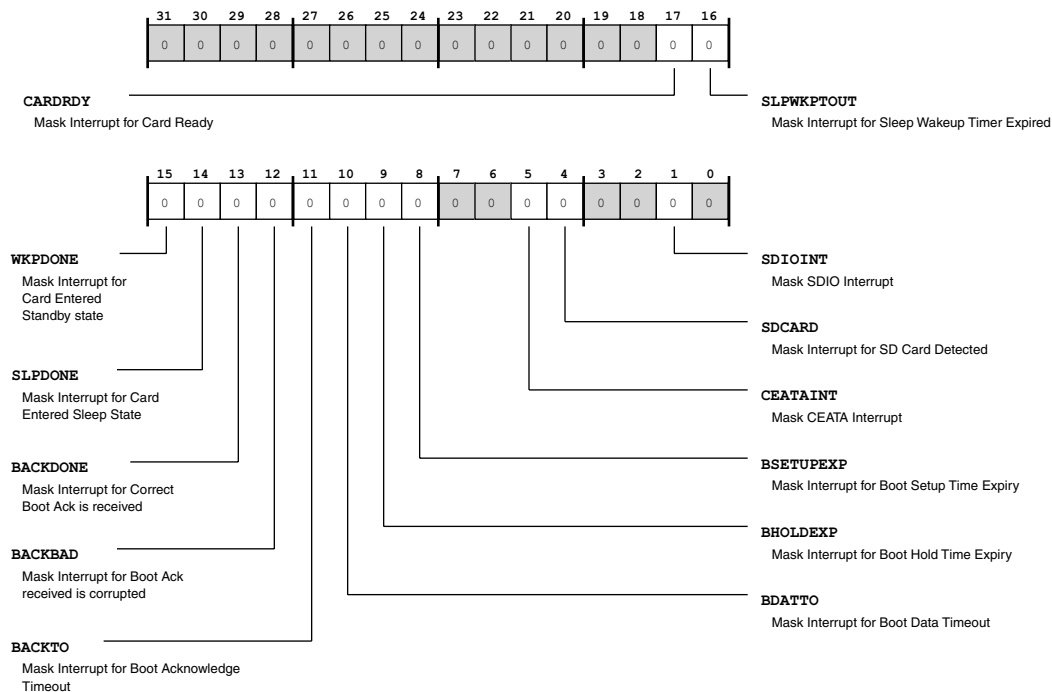


**Figure 24-30:** RSI\_IMSK0 Register Diagram

**Table 24-42:** RSI\_IMSK0 Register Fields

| Bit No. | Bit Name   | Description/Enumeration                            |                   |
|---------|------------|----------------------------------------------------|-------------------|
| 17      | CARDRDY    | Mask Interrupt for Card Ready.                     |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 16      | SLPWKPTOUT | Mask Interrupt for Sleep Wakeup Timer Expired.     |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 15      | WKPDONE    | Mask Interrupt for Card Entered Standby state.     |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 14      | SLPDONE    | Mask Interrupt for Card Entered Sleep State.       |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 13      | BACKDONE   | Mask Interrupt for Correct Boot Ack is received.   |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 12      | BACKBAD    | Mask Interrupt for Boot Ack received is corrupted. |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |
| 11      | BACKTO     | Mask Interrupt for Boot Acknowledge Timeout.       |                   |
|         |            | 0                                                  | Disable Interrupt |
|         |            | 1                                                  | Enable Interrupt  |



**Table 24-42:** RSI\_IMSK0 Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                    |
|---------|-----------|--------------------------------------------|
| 10      | BDATTO    | Mask Interrupt for Boot Data Timeout.      |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |
| 9       | BHOLDEXP  | Mask Interrupt for Boot Hold Time Expiry.  |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |
| 8       | BSETUPEXP | Mask Interrupt for Boot Setup Time Expiry. |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |
| 5       | CEATAINT  | Mask CEATA Interrupt.                      |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |
| 4       | SDCARD    | Mask Interrupt for SD Card Detected.       |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |
| 1       | SDIOINT   | Mask SDIO Interrupt.                       |
|         |           | 0      Disable Interrupt                   |
|         |           | 1      Enable Interrupt                    |

## Configuration Register

RSI\_CFG: Configuration Register - R/W

Reset = 0x0020

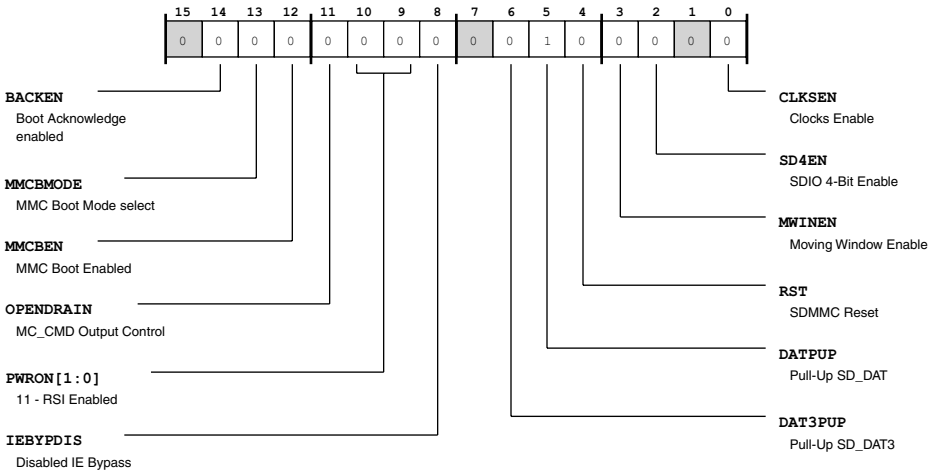


Figure 24-31: RSI\_CFG Register Diagram

Table 24-43: RSI\_CFG Register Fields

| Bit No. | Bit Name  | Description/Enumeration   |                                  |
|---------|-----------|---------------------------|----------------------------------|
| 14      | BACKEN    | Boot Acknowledge enabled. |                                  |
|         |           | 0                         | Boot Acknowledge is not expected |
|         |           | 1                         | Boot Acknowledge is expected     |
| 13      | MMCBMODE  | MMC Boot Mode select.     |                                  |
|         |           | 0                         | Normal Boot Mode                 |
|         |           | 1                         | Alternate Boot Mode              |
| 12      | MMCBEN    | MMC Boot Enabled.         |                                  |
|         |           | 0                         | MMC Boot Not enabled             |
|         |           | 1                         | MMC Boot enabled                 |
| 11      | OPENDRAIN | MC_CMD Output Control.    |                                  |

**Table 24-43:** RSI\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 10:9    | PWRON    | 11 - RSI Enabled.<br>Other combinations reserved   |
|         |          | 0 RSI Disabled                                     |
|         |          | 3 RSI Enabled                                      |
| 8       | IEBYPDIS | Disabled IE Bypass.                                |
|         |          | 0 Input Enable Bypass is not disabled              |
|         |          | 1 Input Enable Bypass is disabled                  |
| 6       | DAT3PUP  | Pull-Up SD_DAT3.                                   |
|         |          | 0 Disables pull-up                                 |
|         |          | 1 Enables pull-up on RSI_DAT[3]                    |
| 5       | DATPUP   | Pull-Up SD_DAT.                                    |
|         |          | 0 Disables pull-up                                 |
|         |          | 1 Enables pull-up on RSI_DAT[2:0] and RSI_DAT[7:4] |
| 4       | RST      | SDMMC Reset.                                       |
|         |          | 0 Normal operation                                 |
|         |          | 1 Reset SDMMC                                      |
| 3       | MWINEN   | Moving Window Enable.                              |
|         |          | 0 Disable moving window                            |
|         |          | 1 Enable moving window                             |
| 2       | SD4EN    | SDIO 4-Bit Enable.                                 |
|         |          | 0 Disable SDIO 4-bit                               |
|         |          | 1 Enable SDIO 4-bit                                |

Table 24-43: RSI\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |                                         |
|---------|----------|-------------------------|-----------------------------------------|
| 0       | CLKSEN   | Clocks Enable.          |                                         |
|         |          | 0                       | Disable RSI clocks (both PCLK and MCLK) |
|         |          | 1                       | Enable RSI clocks (both PCLK and MCLK)  |

Read Wait Enable Register

RSI\_RD\_WAIT: Read Wait Enable Register - R/W

Reset = 0x0000

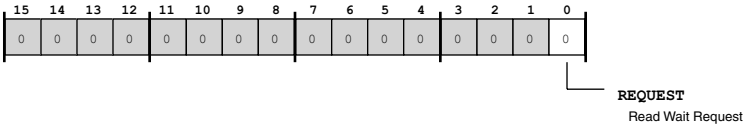


Figure 24-32: RSI\_RD\_WAIT Register Diagram

Table 24-44: RSI\_RD\_WAIT Register Fields

| Bit No. | Bit Name | Description/Enumeration |                                      |
|---------|----------|-------------------------|--------------------------------------|
| 0       | REQUEST  | Read Wait Request.      |                                      |
|         |          | 0                       | Normal operation                     |
|         |          | 1                       | Issue read wait request to SDIO card |

Peripheral Identification Register

RSI\_PID0: Peripheral Identification Register - R/NW

Reset = 0x0000 0000

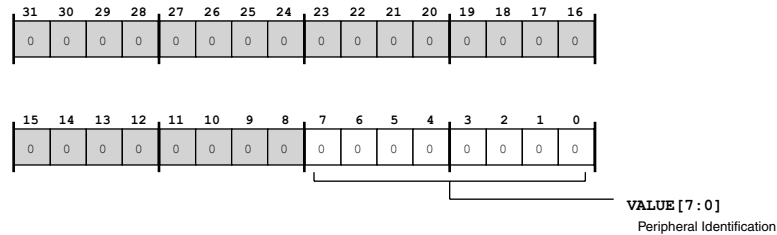


Figure 24-33: RSI\_PID0 Register Diagram

Table 24-45: RSI\_PID0 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 7:0     | VALUE    | Peripheral Identification. |

Peripheral Identification Register

RSI\_PID1: Peripheral Identification Register - R/NW

Reset = 0x0000 0000

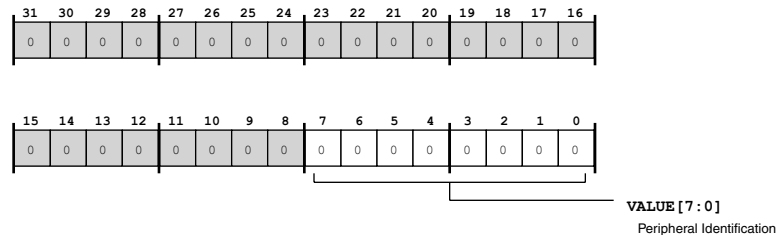


Figure 24-34: RSI\_PID1 Register Diagram

Table 24-46: RSI\_PID1 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 7:0     | VALUE    | Peripheral Identification. |

Peripheral Identification Register

RSI\_PID2: Peripheral Identification Register - R/NW

Reset = 0x0000 0000

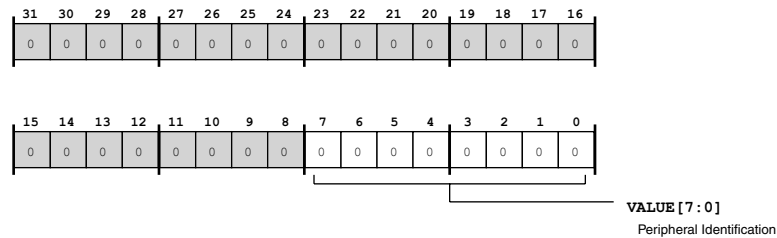


Figure 24-35: RSI\_PID2 Register Diagram

Table 24-47: RSI\_PID2 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 7:0     | VALUE    | Peripheral Identification. |

Peripheral Identification Register

RSI\_PID3: Peripheral Identification Register - R/NW

Reset = 0x0000 0000

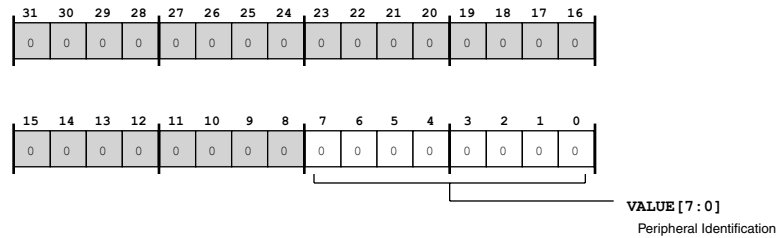


Figure 24-36: RSI\_PID3 Register Diagram

Table 24-48: RSI\_PID3 Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 7:0     | VALUE    | Peripheral Identification. |







# 25 Serial Peripheral Interface (SPI)

The Serial Peripheral Interface is an industry-standard synchronous serial link that supports communication with multiple SPI compatible devices. The baseline SPI peripheral is a synchronous, four-wire interface consisting of two data pins, one device select pin, and a gated clock pin. With the two data pins, it allows for full-duplex operation to other SPI compatible devices. An additional two (optional) data pins are provided to support quad SPI operation. Enhanced modes of operation such as flow control, Fast Mode and dual-I/O mode (DIOM) are also supported. Moreover, a direct memory access (DMA) mode allows for transferring several words with minimal CPU interaction.

The SPI interface includes programmable baud rates, clock phase, and clock polarity. It can operate in a multi-master environment by interfacing with several other devices, acting as either a master device or a slave device. In a multi-master environment, the SPI interface uses open drain outputs to avoid data bus contention.

## SPI Features

The SPI module supports the following features:

- Full-duplex, synchronous serial interface
- Supports 8, 16 and 32-bit word sizes
- Programmable baud rate, clock phase and polarity
- Programmable inter-frame latency
- Flow control
- Support for Fast, DIOM and Quad SPI enhanced modes
- Independent receive and transmit DMA channels
- Burst transfer mode for non-DMA write accesses

## SPI Functional Description

The following sections provide functional descriptions of the SPI:

- [SPI Register List](#)
- [SPI Block Diagram](#)

The SPI is essentially a shift register that serially transmits and receives data bits to/from other SPI devices. During an SPI transfer, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling of the information on the two serial data lines.

During a data transfer, one SPI system acts as the link master which controls the data flow, while the other system acts as the slave, which has data shifted into and out of it by the master. Different devices can take turn being masters, and one master may simultaneously shift data into multiple slaves (broadcast mode). However, only one slave may drive its output to write data back to the master at any given time. This must be enforced in the broadcast mode, where several slaves can be selected to receive data from the master, but only one slave can be enabled to send data back to the master.

SPI supports enhanced modes of operation like Fast Mode, DIOM and Quad-SPI, as well as providing optional flow control. In Fast Mode, received data is sampled on the transmit edge instead of the standard receive edge, thus enabling a full-cycle path for the received data. In DIOM, both MOSI and MISO are configured as input or output pins, and two bits are shifted in or out on each receive or transmit edge. In Quad-SPI mode, SPI\_D3:0 are configured as input or output pins and four bits are shifted in or out on each receive or transmit edge. Flow control can be used by a slower slave to stall a faster master device.

The SPI can be used in a single master as well as multi-master environment. The SPI\_MOSI, SPI\_MISO, and the SPI\_CLK signals are all tied together in both configurations. SPI transmission and reception may be enabled simultaneously or individually, depending on SPI\_RXCTL and SPI\_TXCTL settings. In Broadcast mode, several slaves can be enabled to receive, but only one slaves must be in transmit mode and driving the SPI\_MISO line.

## SPI Register List

The serial peripheral interface SPI provides a full-duplex, synchronous serial interface, which supports both master/slave modes and multi-master environments. The SPI's baud rate and clock phase/polarities are programmable, and it has integrated DMA channels for both transmit and receive data streams. A set of registers govern SPI operations. For more information on SPI functionality, see the SPI register descriptions.

**Table 25-1:** SPI Module Registers

| Name      | Description               |
|-----------|---------------------------|
| SPI_CTL   | Control Register          |
| SPI_RXCTL | Receive Control Register  |
| SPI_TXCTL | Transmit Control Register |
| SPI_CLK   | Clock Rate Register       |

**Table 25-1:** SPI Module Registers (Continued)

| Name         | Description                            |
|--------------|----------------------------------------|
| SPI_DLY      | Delay Register                         |
| SPI_SLVSEL   | Slave Select Register                  |
| SPI_RWC      | Received Word Count Register           |
| SPI_RWCR     | Received Word Count Reload Register    |
| SPI_TWC      | Transmitted Word Count Register        |
| SPI_TWCR     | Transmitted Word Count Reload Register |
| SPI_IMSK     | Interrupt Mask Register                |
| SPI_IMSK_CLR | Interrupt Mask Clear Register          |
| SPI_IMSK_SET | Interrupt Mask Set Register            |
| SPI_STAT     | Status Register                        |
| SPI_ILAT     | Masked Interrupt Condition Register    |
| SPI_ILAT_CLR | Masked Interrupt Clear Register        |
| SPI_RFIFO    | Receive FIFO Data Register             |
| SPI_TFIFO    | Transmit FIFO Data Register            |

## ADSP-BF60x SPI Interrupt List

**Table 25-2:** ADSP-BF60x SPI Interrupt List

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| SPI0 TX DMA Channel | 55 | 6           | LEVEL       |
| SPI0 RX DMA Channel | 56 | 7           | LEVEL       |
| SPI0 Status         | 57 |             | LEVEL       |
| SPI1 TX DMA Channel | 58 | 8           | LEVEL       |

**Table 25-2:** ADSP-BF60x SPI Interrupt List (Continued)

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| SPI1 RX DMA Channel | 59 | 9           | LEVEL       |
| SPI1 Status         | 60 |             | LEVEL       |

## ADSP-BF60x SPI Trigger List

**Table 25-3:** Masters ADSP-BF60x SPI Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| SPI0 TX DMA Channel | 26 | PULSE/EDGE  |
| SPI0 RX DMA Channel | 27 | PULSE/EDGE  |
| SPI1 TX DMA Channel | 28 | PULSE/EDGE  |
| SPI1 RX DMA Channel | 29 | PULSE/EDGE  |
|                     |    |             |

**Table 25-4:** Slaves ADSP-BF60x SPI Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| SPI0 TX DMA Channel | 26 |             |
| SPI0 RX DMA Channel | 27 |             |
| SPI1 TX DMA Channel | 28 |             |
| SPI1 RX DMA Channel | 29 |             |
|                     |    |             |

## SPI Block Diagram

The figure below illustrates the block diagram of the SPI module. The module is comprised of three primary parts:

- SPI core contains the receive and transmit FIFOs and their associated shift registers.
- Control blocks contain the synchronizer and logic to control the data flow through the data pipelines.
- Register block.

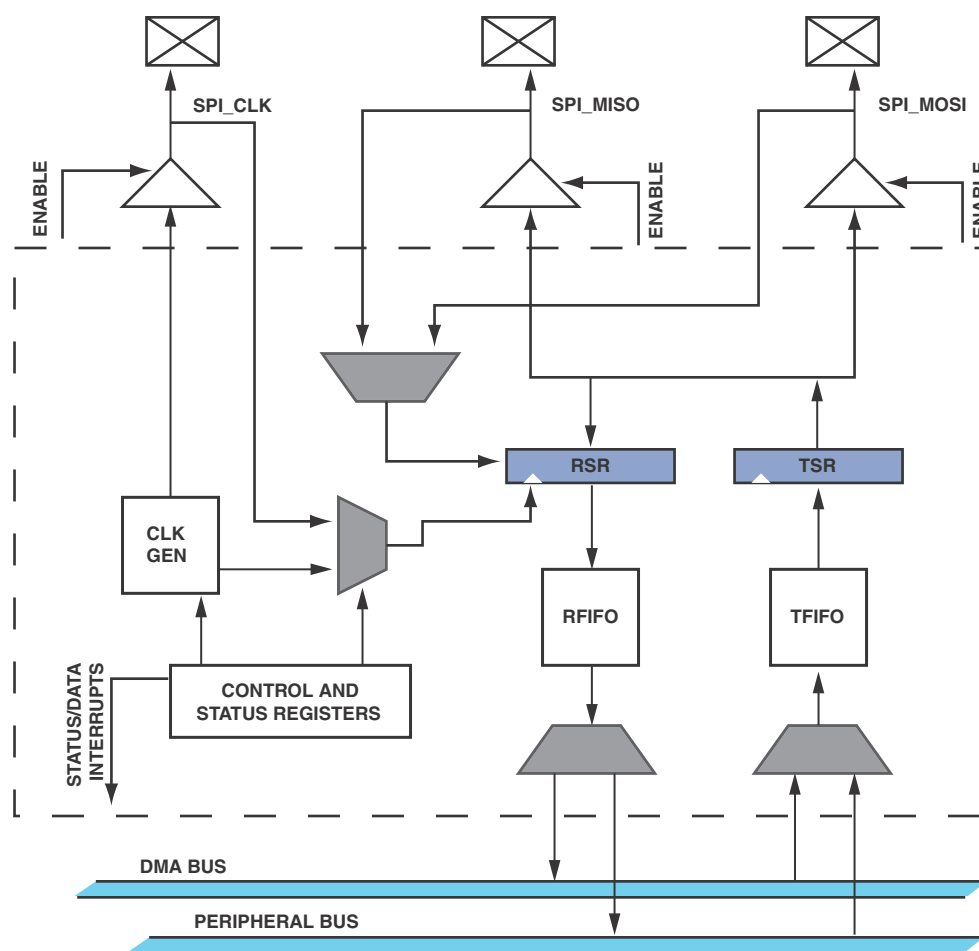


**Figure 25-1:** SPI Controller Block Diagram

## SPI Transfer Protocol

The SPI module implements two channels that are independent of each other. These channels are controlled by their dedicated control registers – SPI\_RX\_CTL and SPI\_TX\_CTL. Except in Dual and Quad modes, both channels may be enabled and used simultaneously.

The SPI protocol supports four different combinations of serial clock phase and polarity. These combinations are selected through the `SPI_CTL.CPOL` and `SPI_CTL.CPHA` bits.

The figures below demonstrate the two basic transfer formats as defined by the `CPHA` bit. Two waveforms are shown for `SPI_CLK`—one for `CPOL = 0` and the other for `CPOL = 1`. The diagrams may be interpreted as master or slave timing diagrams since the `SPI_CLK`, `SPI_MISO` and `SPI_MOSI` pins are directly connected between the master and the slave. The `SPI_MISO` signal is the output from the slave (slave transmission), and the `SPI_MOSI` signal is the output from the master (master transmission). The `SPI_CLK` signal is generated by the master, and the `SPI_SS` signal is the slave device select input to the slave from the master. The diagrams represent an 8-bit transfer (`SPI_CTL.SIZE = 0`) with the MSB first (`SPI_CTL.LSBF = 0`). Any combination of the `SIZE` and `LSBF` bits of `SPI_CTL` is allowed. For example, a 16-bit transfer with the LSB first is another possible configuration.

The clock polarity and the clock phase should be identical for the master device and the slave device involved in the communication link. The transfer format from the master may be changed between transfers to adjust to various requirements of a slave device.

The `SPI_CTL.ASSEL` bit determines if the `SPI_SS` line is controlled by the SPI hardware or by software. When `ASSEL = 1`, the slave select line must be set to the polarity set in the `SPI_CTL.SELST` field between each serial transfer. The actual behavior of `SPI_SS` would also depend on the parameters programmed into the `SPI_DLY` register. This is controlled automatically by the SPI hardware logic. When `ASSEL = 0`, `SPI_SS` may either remain active between successive transfers or be inactive. This must be controlled by the software via manipulation of the `SPI_SSEL` register.

The figures below illustrate the case when `ASSEL = 1` and the `SPI_SS` line is inactive between frames. If `ASSEL = 0`, the `SPI_SS` line may remain active between frames; however, the first bit will only be driven when an active transition of `SPI_CLK` occurs.

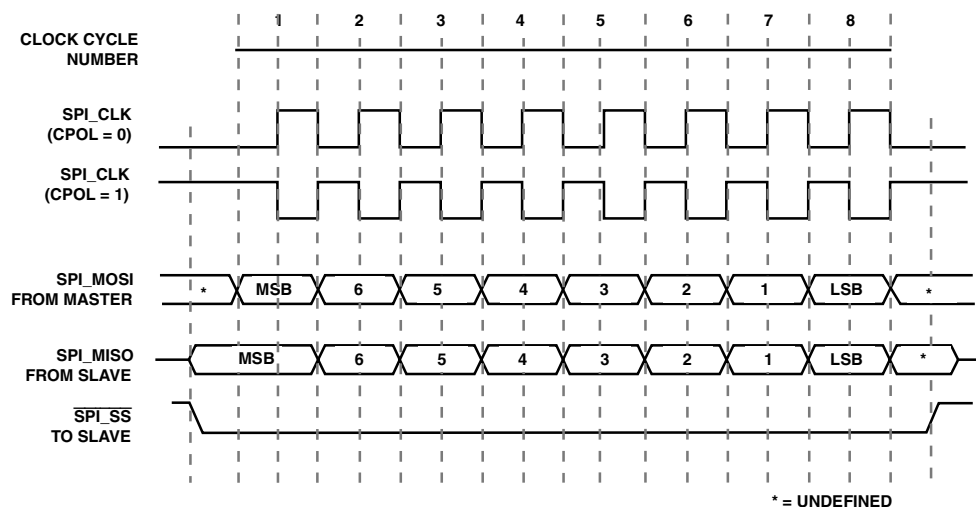


**Figure 25-2:** SPI Transfer Protocol for `CPHA=0`

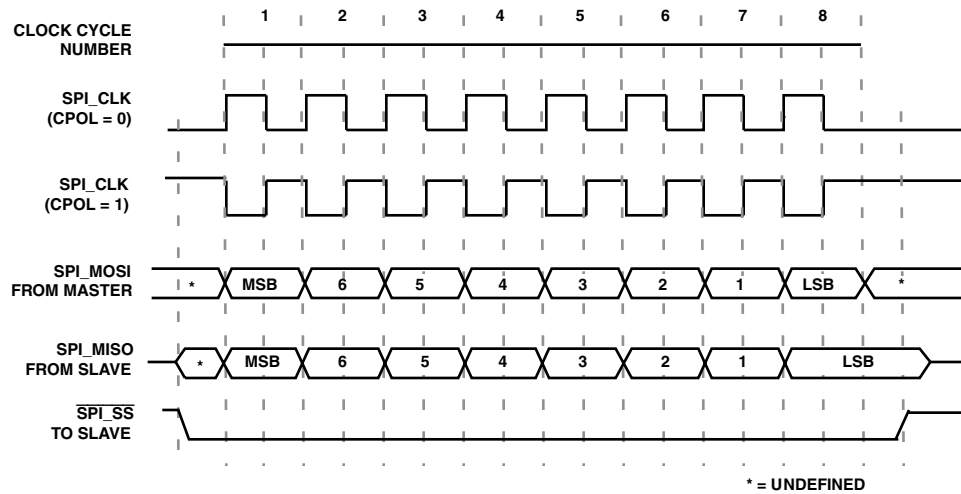


**Figure 25-3:** SPI Transfer Protocol for CPHA=1

## SPI Clock Considerations

The `SPI_CLK` signal is a gated clock that is only active during data transfers, for the duration of the transferred word. In normal mode, the number of active edges is equal to the number of bits to be transmitted or received. In dual-I/O mode it is half of the number of bits to be transmitted or received, and in quad-SPI mode it is one-fourth of the number. The clock rate can be as high as the `SCLK` rate, and both even and odd dividers from `SCLK` are supported. For master devices, the clock rate is determined by the `SPI_CLK` register value, whereas this value is ignored for slave devices.

When the SPI controller is a master, `SPI_CLK` is an output signal. Conversely, when the SPI controller is a slave, `SPI_CLK` is an input signal. Slave devices ignore the SPI clock if the slave select input is driven inactive. The `SPI_CLK` signal is used to shift out and shift in the data driven onto the `SPI_MISO` and `SPI_MOSI` lines. The data is always shifted out on one edge of the clock (the active edge) and sampled on the opposite edge of the clock (the sampling edge). Clock polarity and clock phase relative to data are programmable through the `SPI_CTL` register and define the transfer format.

## Controlling Delay Between Frames

The figure below illustrates SPI timing with `SPI_DLY.LEADX` and `SPI_DLY.LAGX` programming. The `LAGX` controls the timing between the Slave Select (`SPI_SEL`) assertion and the first `SPI_CLK` edge, while `LEADX` controls the timing between the last `SPI_CLK` edge and deassertion of `SPI_SEL`. The lead and lag timing can be extended by 1 `SPI_CLK` duration to ease timing restrictions on the slave device.

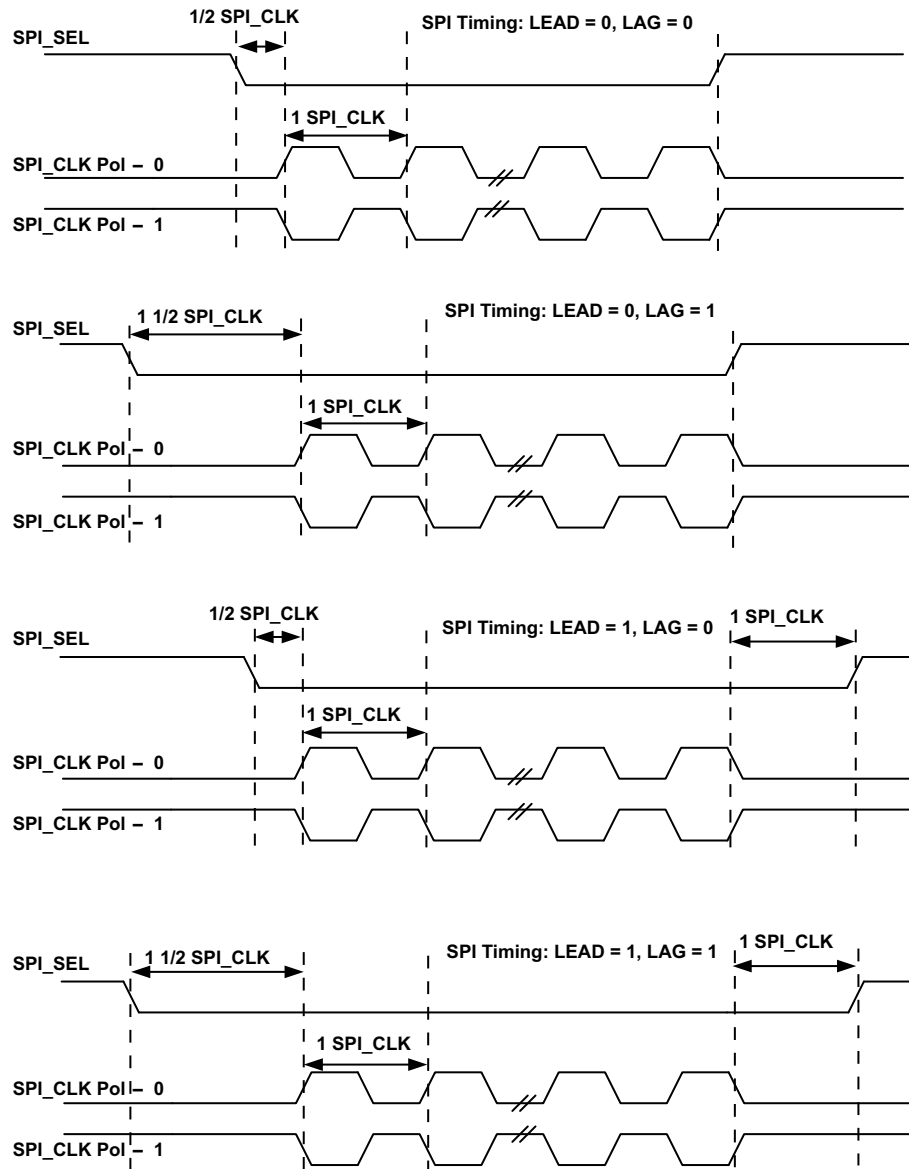


**Figure 25-4:** SPI Timing with Lead and Lag Programming (independent of SPI\_CTL.CPHA setting)

The figure below illustrates SPI timing with STOP programming which is used to insert multiples of SPI\_CLK period delays between transfers. The SPI\_SEL line is deasserted for the duration specified in the SPI\_DLY.STOP field, assuming the SPI\_CTL.SELST bit is configured for deassertion between transfers.

If SPI\_DLY.STOP is programmed to zero, the master operates in a *continuous mode*, resulting in immediate start of the second word after the last bit is transferred from the first word. During this mode of operation, the slave select line is continuously asserted.



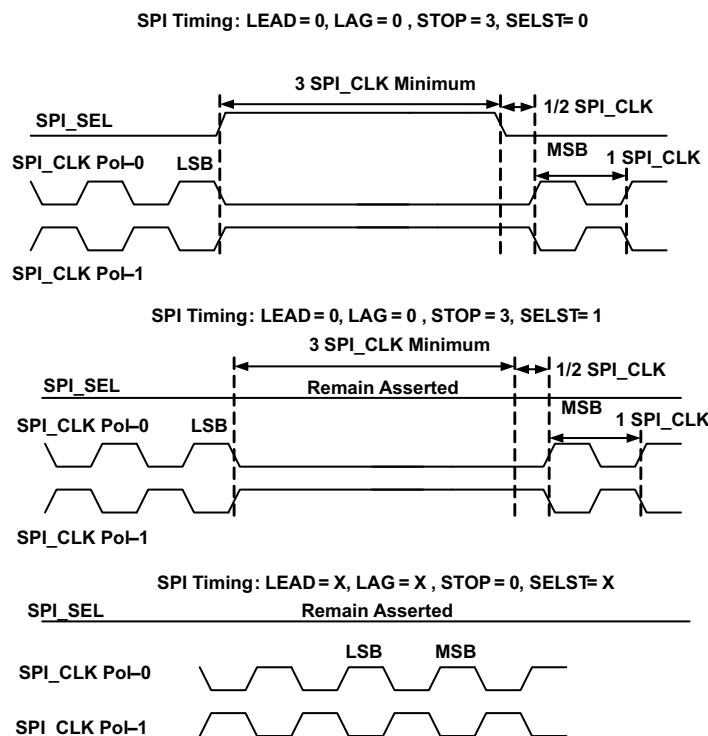


**Figure 25-5:** SPI Timing with SPI\_DLY.STOP Programming (Independent of SPI\_CTL.CPHA Setting)

When `SPI_DLY.STOP` is zero and initial conditions for a transfer are not met, the interface pauses before the next transfer. During this pause, the state of the slave select pin is determined by the `SPI_CTL.SELST` bit, and the `SPI_DLY.LEAD` and `SPI_DLY.LAG` bits determine the timing between `SPI_CLK` edges and the slave select line.

## SPI Flow Control

In Master mode, the `SPI_RDY` pin acts as an input signal and should be driven by the slave device. `SPI_RDY` can be deasserted by the slave to stop the master from initiating any new transfer. If `SPI_RDY` is deasserted in the middle of a transfer, the current transfer will continue, and the next transfer will not start unless the slave asserts the `SPI_RDY` signal. Whenever the slave deasserts `SPI_RDY` and stalls the master, the SPI controller goes into a waiting state, and the `SPI_STAT.FCS` bit is set. When the slave asserts `SPI_RDY`, the SPI controller resumes operation, and the `SPI_STAT.FCS` bit is cleared.

In Slave mode, the `SPI_RDY` pin acts as an output signal. Flow control can be configured on either the TX channel or the RX channel. This is controlled by the `SPI_CTL.FCCH` bit. If flow control is configured on the TX channel, as the `SPI_TFIFO` status nears the empty condition, `SPI_RDY` is deasserted. If flow control is configured on the RX channel, as the `SPI_RFIFO` status nears the full condition, `SPI_RDY` is deasserted. The FIFO status at which `SPI_RDY` deassertion should take place can be controlled by the `SPI_CTL.FCWM` bits. Note that flow control in Slave mode is purely based on the FIFO status and doesn't depend on the word counters.

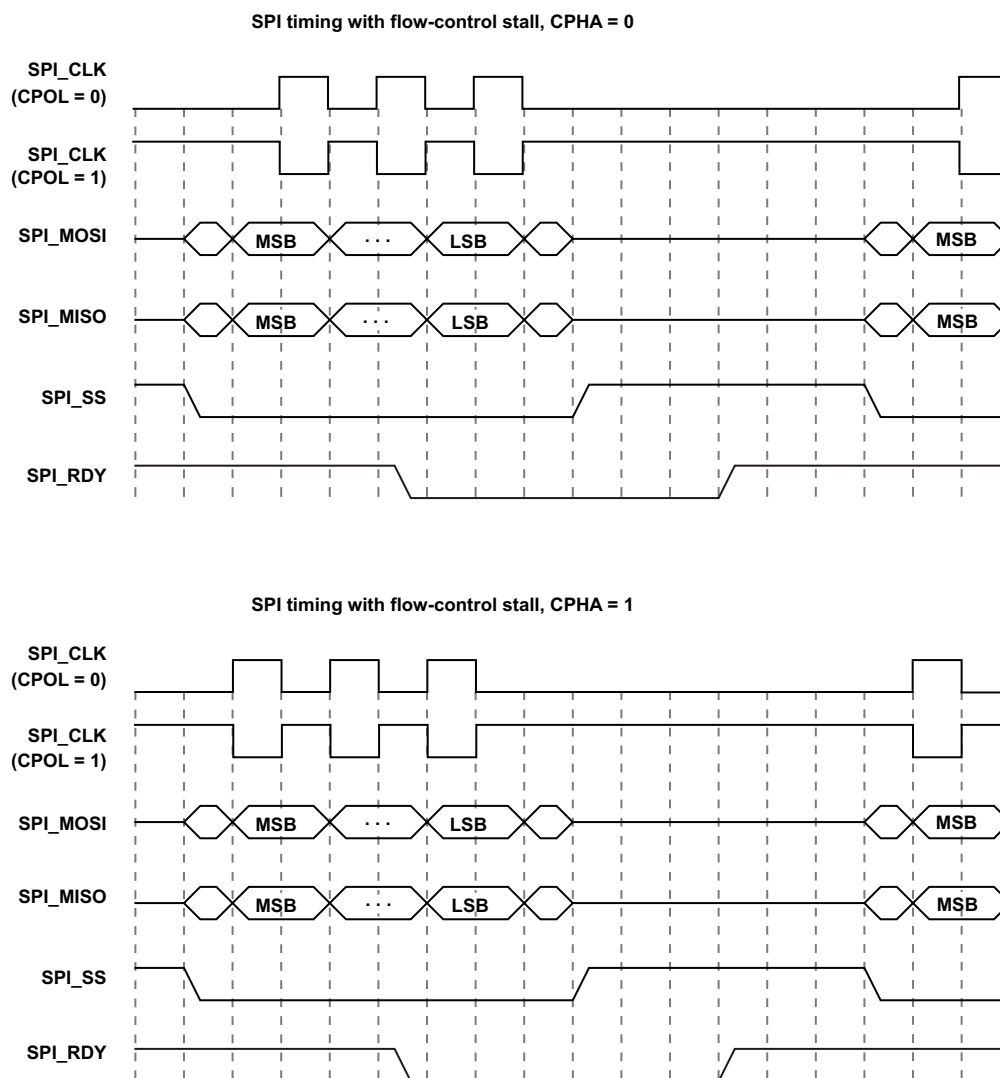


**Figure 25-6:** SPI Flow Control Timing in Master Mode.

## SPI Slave Select Operation

If the SPI is in Slave mode,  $\overline{\text{SPI\_SS}}$  acts as the slave select input. When SPI is enabled as a master,  $\overline{\text{SPI\_SS}}$  can serve as an error detection input for the SPI in a multi-master environment. The `SPI_CTL.PSSE` bit enables this feature. When `PSSE = 1`, the  $\overline{\text{SPI\_SS}}$  input is the master mode error input. Otherwise,  $\overline{\text{SPI\_SS}}$  is ignored.

The  $\overline{\text{SPI\_SS}}$  signal is an active-low signal and should be asserted during the transfer by the master. It can be deasserted or remain asserted between transfers. When  $\overline{\text{SPI\_SS}}$  is deasserted, `SPI_CLK` and inputs are ignored, and outputs are three-stated.

The `SPI_SLVSEL.SSEX` bits are used in a multiple-slave SPI environment. For example, if there are eight SPI devices in the system including a processor master, the master processor can support the SPI mode transactions across the other seven devices. This configuration requires only one master processor in this multi-slave environment.

For example, assume that the processor's SPI is the master. `SPI_SSEL1` to `SPI_SSEL7` on the processor can be connected to the slave select pin of each slave device. In this configuration, the `SPI_SLVSEL.SSEX` bits can be used in three ways. In cases 1 and 2, the processor is the master and the seven micro controllers/peripherals with SPI interfaces are slaves. The processor can do one of the following:

1. Transmit to all seven SPI devices at the same time in a broadcast mode. Here, all `SPI_SLVSEL.SSEX` bits are set.
2. Receive and transmit from one SPI device by enabling only one slave SPI device at a time.
3. If all the slaves are also processors, then the requester can receive data from only one processor (enabled by clearing the `SPI_CTL.EMISO` bit in the six other slave processors) at a time and transmit broadcast data to all seven at the same time. This EMISO feature may be available in some other micro controllers. Therefore, it is possible to use the EMISO feature with any other SPI device that includes this functionality.

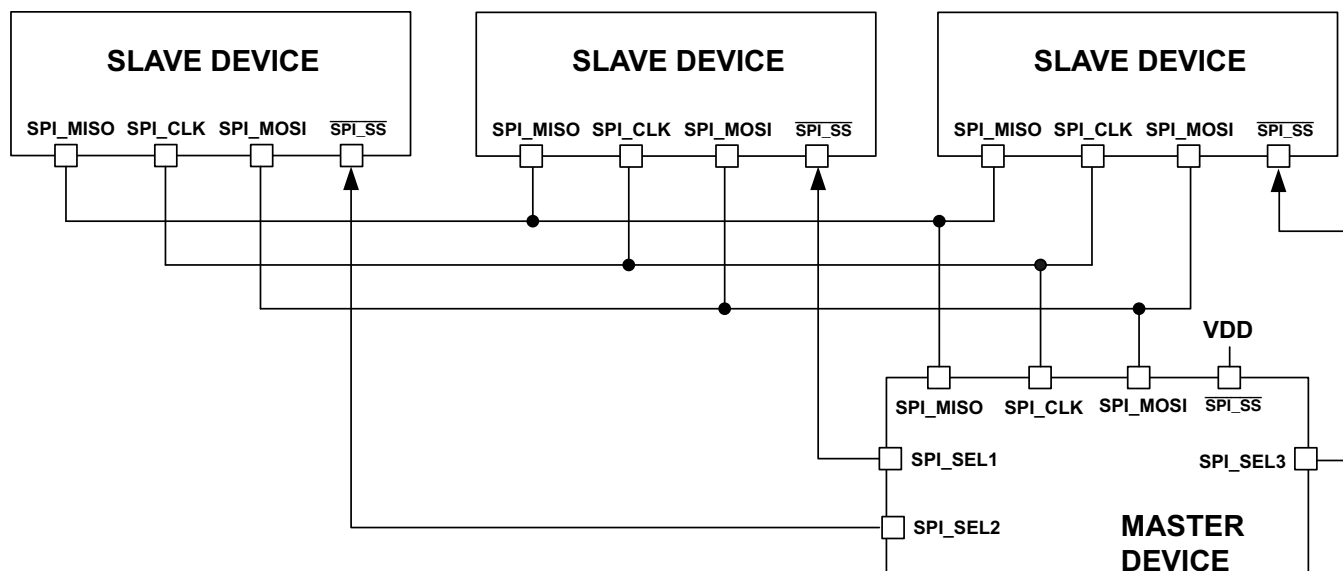


**Figure 25-7:** Single-Master, Multiple-Slave Configuration

## Beginning and Ending a Non-DMA SPI Transfer

The start and finish of a non-DMA SPI transfer depend on the following settings.

1. Whether the device is configured as a master or a slave
2. The state of the `ASSEL` bit, which selects between hardware and software control over `SSEL`

When `SPI_CTL.CPHA = 0`, the enabled slave select outputs are driven active. However, the `SPI_CLK` signal remains inactive for the first half of the first cycle of `SPI_CLK`. For a slave with `SPI_CTL.CPHA = 0`, the transfer starts as soon as the  $\overline{\text{SPI\_SS}}$  input goes low.

When `SPI_CTL.CPHA = 1`, a transfer starts with the first active edge of `SPI_CLK` for both slave and master devices. For a master device, a transfer is considered finished after it sends the last data and simultaneously receives the last data bit. A transfer for a slave device ends after the last sampling edge of `SPI_CLK`. If `SPI_CTL.ASSEL = 0`, the hardware maintains responsibility for toggling  $\overline{\text{SPI\_SS}}$  between frames. If `SPI_CTL.ASSEL = 1`, software controls the  $\overline{\text{SPI\_SS}}$  line and may keep it active between frames.

The `SPI_STAT.RFE` bit defines when the receive buffer can be read, indicating that `SPI_RFIFO` is not empty. The `SPI_STAT.TFF` bit defines when the transmit buffer can be written, indicating that the `SPI_TFIFO` is not full. The end of a single word transfer occurs when the `SPI_STAT.RFE` bit is cleared, indicating that a new word has just been received and written into the receive FIFO. `SPI_STAT.RFE` remains cleared as long as the receive FIFO has valid data.

To maintain software compatibility with other SPI devices, the `SPI_STAT.SPIF` bit is also available for polling. This bit may have a slightly different behavior from that of other commercially available devices.

In master mode with the `SPI_CTL.ASSEL` bit cleared, software should manually assert the required slave select signal before starting the transaction. After all data has been transferred, software typically releases the slave select line.

When the receive or transmit word counters are enabled in `SPI_TXCTL` or `SPI_RXCTL`, a finish interrupt is generated at the end of the transfer to signal the end of all transfers relating to that transaction.

## **SPI Transmit Operation in Non-DMA Mode**

Transmit operation on non-DMA mode is enabled through the `SPI_TXCTL.TEN` bit. Transmit operation can be enabled independently from receive operation, and the transmit channel can become the initiating channel based on the `SPI_TXCTL.TTI` setting.

Transmit underrun is not possible in this mode, as no new transfer would be initiated unless the transmit FIFO is empty (in the case that `SPI_TXCTL.TTI = 1`). A receive overflow is detected when data from a new frame transfer replaces older data in a full receive FIFO. This can occur if `SPI_TXCTL.TTI = 1` and the receive channel is enabled in a non-initiating capacity.

An SPI transmit interrupt is signalled once the transmit channel has been enabled and the transmit FIFO is not full. The frequency of the interrupt is controlled by the `SPI_TXCTL.TDR` setting.

## **SPI Receive Operation in Non-DMA Mode**

Receive operation on non-DMA mode is enabled through the `SPI_RXCTL.REN` bit. Receive operation can be enabled independently from transmit operation, and the receive channel can become the initiating channel based on the `SPI_RXCTL.RTI` setting.

Receive overflow is not possible in this mode, as no new transfer would be initiated when the receive FIFO is full (in the case of `SPI_RXCTL.RTI = 1`). A transmit underrun can occur (`SPI_TXCTL.TDU`) if no valid data is in the `SPI_TFIFO` register upon initiation of a transfer. This can occur if `SPI_RXCTL.RTI = 1` and the transmit channel is enabled in a non-initiating capacity.

A SPI receive interrupt is signaled once the receive channel has been enabled and there is data waiting to be read. The frequency of the interrupt is controlled by the `SPI_RXCTL.RDR` setting.

## SPI Dual I/O Mode

In Dual I/O mode, the `SPI_MISO` and `SPI_MOSI` pins are configured to operate in the same direction which doubles bandwidth. The order of bits on the pins are determined by the `SPI_CTL.SOSI` bit which, when set, sends the first bit on the `SPI_MOSI` pin and the second bit on the `SPI_MISO` pin. If the `SPI_CTL.SOSI` bit is cleared, the order is reversed. Since Dual I/O mode uses both pins to transmit or receive data, only one channel can be enabled, either transmit or receive. Flow control through the `SPI_RDY` pin is supported. Interrupt generation is unaffected by Dual I/O mode. However, the gap between successive interrupts is reduced, since the individual transfer latency is halved.

Changing to Quad SPI mode should be done when the SPI is in a quiescent state.

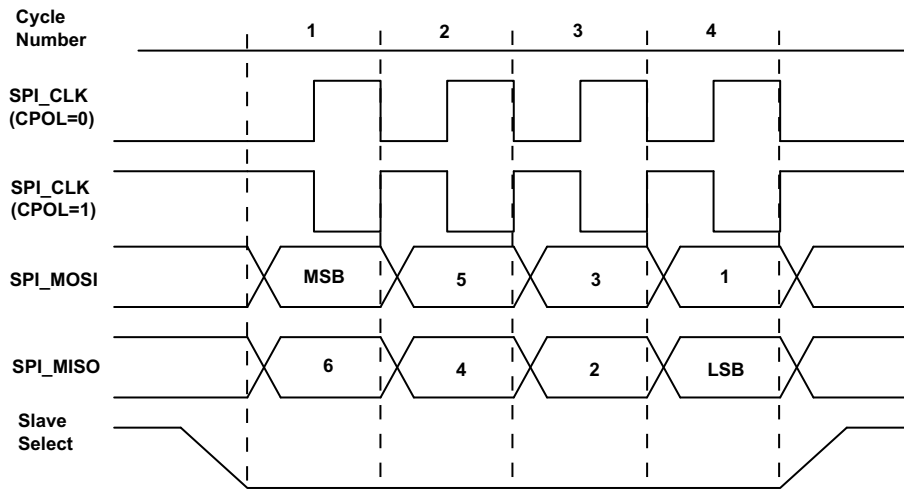


**Figure 25-8:** Dual I/O Mode Transfer Protocol for `CPHA=0`, `SOSI=1`, 8-Bit Transfer, `LSBF=0`.

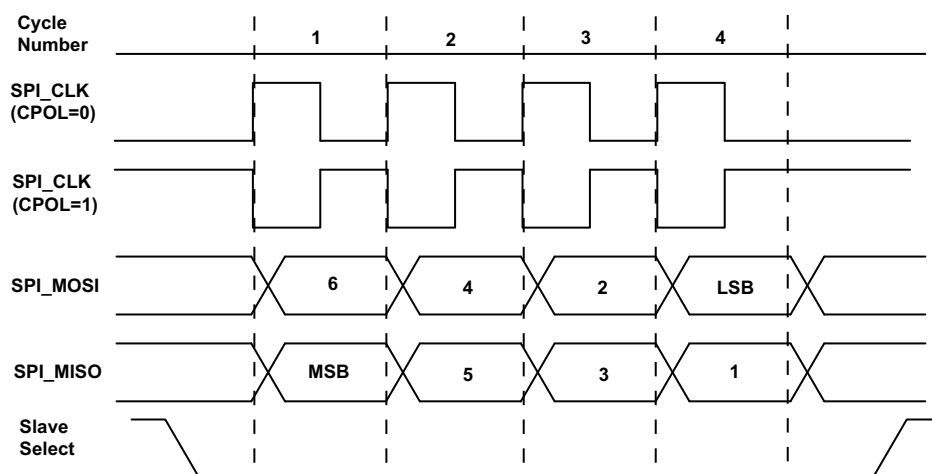


**Figure 25-9:** Dual I/O Mode Transfer Protocol for CPHA=1, SOSI=0, 8-Bit Transfer, LSBF=0.

## SPI Quad I/O Mode

In Quad SPI mode, the SPI\_MISO and SPI\_MOSI pins, in tandem with the SPI\_D2 and SPI\_D3 pins, are configured to operate in the same direction. The order of bits on the pins are determined by the SPI\_CTL.SOSI bit which, when set, sends the first bit on the SPI\_MOSI pin, the second bit on the SPI\_MISO pin, the third bit on the SPI\_D2 pin and the fourth bit on the SPI\_D3 pin. If the SPI\_CTL.SOSI bit is cleared, the order is reversed. Since Quad SPI mode uses all four pins to transmit or receive data, only one channel can be enabled, either Transmit or Receive. Flow control through the SPI\_RDY pin is supported. Interrupt generation is unaffected by Quad SPI mode.

Changing to Quad SPI mode should be done when the SPI is in a quiescent state.

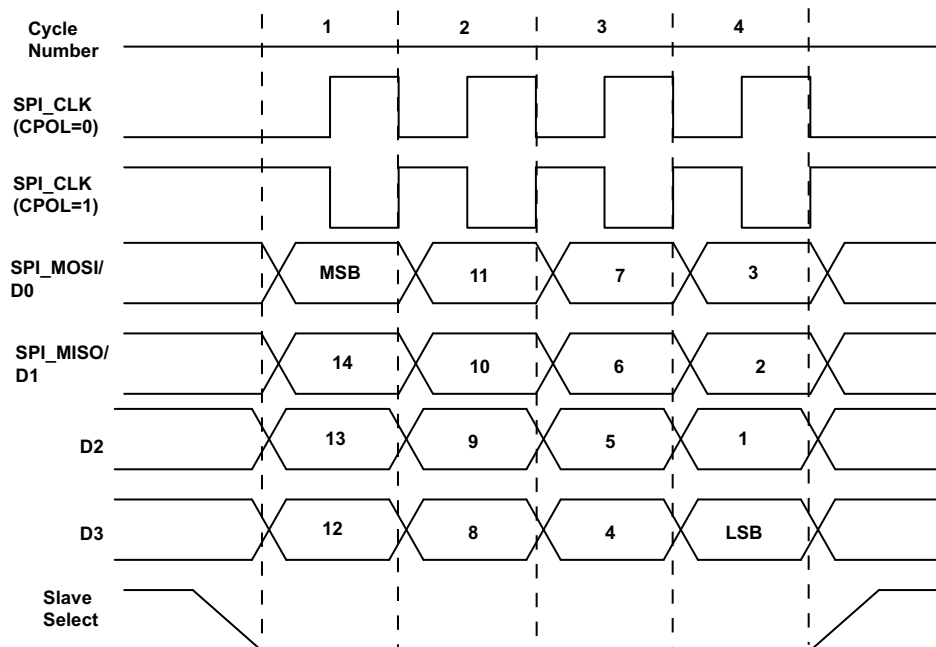


**Figure 25-10:** Quad Mode Timing for CPHA=0, SSI=1, 16-Bit Transfer, LSBF=0.

**NOTE:** Quad SPI 8-bit transfer is not supported in Slave Continuous mode of operation with a SCLK:SPI\_CLK ratio less than 1:2. A minimum of 2 SCLK cycles is required between transfers in 8-bit Quad SPI Slave mode with SCLK:SPI\_CLK ratio less than 1:2.

## SPI Fast Mode

Fast Mode is similar to normal mode of operation when transmitting. When receiving, data is sampled at the next transmit edge allowing a full cycle of timing in the receive direction. This mode is valid in master mode operation only. When the SPI is operating in Fast Mode, the slave should drive the data for one full cycle.

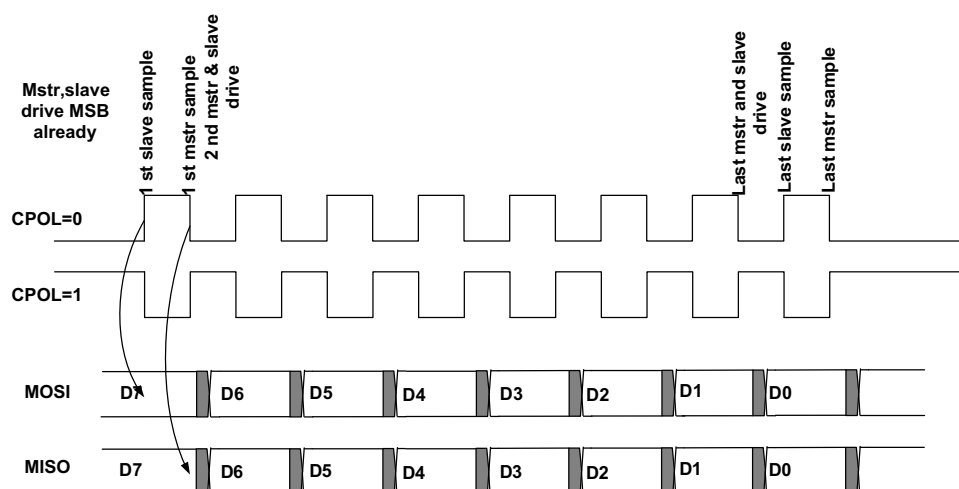


**Figure 25-11:** SPI Transfer Protocol in Fast Mode for SPI\_CTL.CPHA = 0

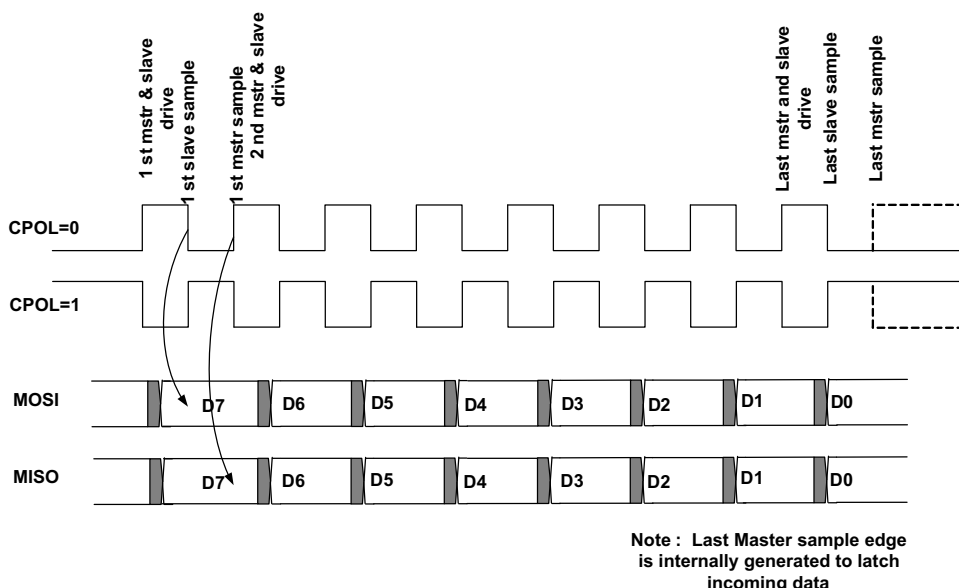


**Figure 25-12:** SPI Transfer Protocol in Fast Mode for SPI\_CTL.CPHA = 1

## SPI Interrupt Signals

The SPI controller supports three types of interrupt signals, corresponding to data, status, and error conditions.

## SPI Data Interrupts

The SPI peripheral supports two data interrupt channels – receive and transmit. These interrupt signals are multiplexed into the DMA request lines. Since the peripheral interfaces to independent read and write



interfaces with DMA, the read and write data interrupts are independent. When the DMA channel(s) are not being used, the interrupts are routed directly to the system event controller, occupying the same interrupt vector locations as the corresponding DMA channels do.

Each of the data interrupts can be individually controlled by programming the `SPI_RXCTL.RDR` and `SPI_TXCTL.TDR` fields for receive and transmit, respectively. When RX is enabled, the RX interrupt is issued whenever there is data available in the receive data path to be read (according to the `SPI_RXCTL.RDR` setting). When TX is enabled, the TX interrupt is issued whenever the transmit data path can be written to (according to the `SPI_TXCTL.TDR` setting). DMA data interrupts are made compatible with second generation DMA to incorporate urgent data request & transfer finish interrupt apart from usual data request interrupt. Note that transmit interrupts operate independently from the word counter value in `SPI_TWC`.

## SPI Status Interrupts

The SPI controller supports several status interrupts to indicate different conditions of the receiver and transmitter. All status interrupts can be masked. Status interrupts are signaled directly through a single SPI Status IRQ line, which may or may not be combined with the SPI Error IRQ line for a given processor. The following table describes the status interrupts that are available for the SPI controller.

**Table 25-5:** SPI Status Interrupts

| SPI_STAT Bit | Description                                                                                                                                                                                                                                                                                                                                            |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RUWM         | Receive FIFO Urgent Watermark Interrupt. Issued when the level of the RFIFO breaches the watermark set in the <code>SPI_RXCTL.RUWM</code> field. It is cleared when the level of the RFIFO reaches the watermark set in the <code>SPI_RXCTL.RRWM</code> field. If the RX channel is configured in DMA mode, RUWM is multiplexed with the data request. |
| TUWM         | Receive FIFO Urgent Watermark Interrupt. Issued when the level of the TFIFO breaches the watermark set in <code>SPI_TXCTL.TUWM</code> . It is cleared when the level of the TFIFO reaches the watermark set in the <code>SPI_TXCTL.TRWM</code> field. If the TX channel is configured in DMA mode, TUWM is multiplexed with the data request.          |
| TS           | Transmit Start Interrupt. Issued when the start of a transmit burst is detected by loading of <code>SPI_TWC</code> with the contents of <code>SPI_TWCR</code> .                                                                                                                                                                                        |
| RS           | Receive Start Interrupt. Issued when the start of a receive burst is detected by loading of <code>SPI_RWC</code> with the contents of <code>SPI_RWCR</code> .                                                                                                                                                                                          |
| TF           | Transmit Finish Interrupt. Issued when a transmit burst completes ( <code>SPI_TWC</code> decrements to zero).                                                                                                                                                                                                                                          |
| RF           | Receive Finish Interrupt. Issued when a receive burst completes ( <code>SPI_RWC</code> decrements to zero).                                                                                                                                                                                                                                            |

## SPI Error Conditions

The SPI Controller supports interrupts upon several different error conditions. All interrupts are maskable. The individual interrupt indications combine into a single SPI Error IRQ signal, which may be multiplexed on some processors with the aggregated SPI Status IRQ signal. The following table details the possible error indications.

Error conditions and interrupts arise depending on which of the channels (transmit and/or receive) are enabled. If a channel is disabled, all errors relating to it are ignored. When both channels are enabled, errors and interrupts from both channels are enabled.

**Table 25-6:** SPI Error Interrupts

| SPI_STAT Bit | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MF           | Mode Fault. Signalled when another device is also trying to be a master in a multi-master system and drives the SPI_SS input low. This error will be signalled in the Master mode of operation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| TUR          | Transmission Error. Signalled when an underflow condition occurs on the transmit channel. This occurs when a new transfer starts but SPI_TFIFO is empty. This error will not occur in Master Transmit Initiating mode since SPI_TFIFO Not Empty is one of the conditions for transfer initiation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| ROR          | Reception Error. Signalled when an overflow condition occurs on the receive channel. This occurs when a new data word is received, but the SPI_RFIFO is full. This error condition will not occur in Master Receive Initiating mode since SPI_RFIFO Not Full is one of the conditions for transfer initiation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| TC           | Transmit Collision Error. Signalled when loading data to the transmit shift register happens near the first transmitting edge of SPI_CLK. In Slave mode of operation, the SPI controller is unaware of when the next transfer will start, so loading of data to the transmit shift register may happen just after the transmitting edge. This will result in setup time not being met for the first bit being transmitted, and thus the transmitted data will be corrupted. In SPI_CTL.CPHA = 1 mode, the first SPI_CLK edge is taken as first transmitting edge, whereas if SPI_CTL.CPHA=0 the last SPI_CLK edge of the last transmission (SPI_CTL.SELST=1) or slave select deassertion (SPI_CTL.SELST=0) is taken as the first transmitting edge. This error is signalled only in Slave mode of operation. In Master mode of operation, it is always ensured that loading of data happens before the first transmitting edge of SPI_CLK. |

## SPI Programming Concepts

The following sections provide general programming guidelines and procedures.

## Programming Guidelines

It is acceptable to program `SPI_RX_CTL` and `SPI_TX_CTL` registers after programming `SPI_CTL`, but the initiating mode register and its counter register, if enabled, should be programmed after the non-initiating mode register. For example, if Transmit is the initiating mode and Receive is the non-initiating mode, then `SPI_RX_CTL` and `SPI_RWC` should be programmed before `SPI_TX_CTL` and `SPI_TWC`. If both transmit and receive are to be enabled in initiating mode, `SPI_CTL` should be enabled after programming both `SPI_RX_CTL` and `SPI_TX_CTL`.

These programming guidelines prevent SPI from starting a transfer when SPI registers are still being programmed. Other ways of programming are also allowed as long as commencement of communication is prevented by initiating conditions until all the utilized SPI registers are programmed.

Precautions must be taken to avoid data corruption when changing the SPI module configuration. The configuration must not be changed during a data transfer. Additionally, the clock polarity should only be changed when no slaves are selected. An exception to this is when a SPI communication link consists of a single master and a single slave, `SPI_CTL.ASSEL = 0`, and the slave select input of the slave is permanently tied low. In this case, the slave is always selected, and data corruption can be avoided by enabling the slave only after both the master and slave devices are configured.

The module supports 8, 16 and 32-bit word sizes. To ensure correct operation, both the master and slave must be configured with the same word size.

## SPI Master Operation in Non-DMA Modes

This section describes the operation of the SPI as a master in non-DMA mode.

1. Write to `SPI_SLVSEL`, setting one or more of the SPI Select Enable bits (`SSEx`). This ensures that the desired slaves are properly deselected while the master is configured.
2. The `SPI_RXCTL.RTI` and `SPI_TXCTL.TTI` bits determine the SPI initiating mode. The initiating mode defines the primary transfer channel, and also the initiating condition for the transfer.
3. Write to the `SPI_CLK`, `SPI_CTL`, `SPI_RXCTL` and `SPI_TXCTL` registers, enabling the device as a master and configuring the SPI system by specifying the transfer modes and channels, appropriate word length, transfer format, baud rate, and other control information.

*ADDITIONAL INFORMATION:* If `SPI_RXCTL.RTI` is enabled and `SPI_TXCTL.TTI` is not, write to the `SPI_RXCTL` after writing into `SPI_CTL`, `SPI_TXCTL` and `SPI_TFIFO` to prevent a transmit underrun for the first transfer.

4. If `SPI_CTL.ASSEL = 0`, the user activates the desired slaves by clearing one or more of the `SPI_SLVSEL` flag bits. Otherwise, the SPI hardware takes care of slave activation.
5. The SPI controller then generates the programmed clock pulses on `SPI_CLK` and simultaneously shifts data out of `SPI_MOSI` while shifting data in from `SPI_MISO`. Before a shift, the shift register is loaded

with the contents of the SPI\_TFIFO register. At the end of the transfer, the contents of the shift register are loaded into SPI\_RFIFO.

6. Whenever the initiating conditions are satisfied, the SPI continues to send and receive words. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI\_TXCTL.TDU and SPI\_RXCTL.RDO bits.
7. It is possible to program a secondary channel in addition to the initiating channel. This feature allows the user to utilize the unused channel resources for receives or transmits simultaneously with the initiating channel.

## **SPI Slave Operation in Non-DMA Modes**

When a device is enabled as a slave in a non-DMA mode, the start of a transfer is triggered by a transition of the  $\overline{\text{SPI\_SS}}$  select signal to the active state (low), or by the first active edge of SPI\_CLK, depending on the state of SPI\_CTL.CPHA. The interface operates in the following manner:

1. The core writes to the SPI\_CTL, SPI\_RXCTL and SPI\_TXCTL registers to define the mode of the serial link to be the same as the mode setup in the SPI master.
2. To prepare for the data transfer, the core writes data to be transmitted into SPI\_TFIFO.
3. Once the  $\overline{\text{SPI\_SS}}$  falling edge is detected, the slave starts sending data on active SPI\_CLK edges and sampling data on inactive SPI\_CLK edges.
4. Reception/transmission continues until  $\overline{\text{SPI\_SS}}$  is released or until the slave has received the proper number of clock cycles.
5. The slave device continues to receive/transmit with each new falling edge transition on  $\overline{\text{SPI\_SS}}$  and/or active SPI\_CLK edge. If the transmit buffer remains empty or the receive buffer remains full, the device operates according to the states of the SPI\_TXCTL.TDU and SPI\_RXCTL.RDO bits.

## **Configuring SPI DMA Master Mode**

The SPI interface supports a write DMA channel and a read DMA channel. These may be used individually or in a lock-step manner in duplex mode (SPI\_TXCTL.TTI= SPI\_RXCTL.RTI=1)

1. Write to the appropriate DMA registers to enable the SPI DMA Channel and to configure the necessary work units, access direction, word count, and so on.
2. Write to the SPI\_SLVSEL register, setting one or more of the SPI flag select bits.
3. Write to the SPI\_CLK and SPI\_CTL registers, enabling the device as a master and configuring the SPI system by specifying the appropriate word length, transfer format, baud rate, etc.
4. Write to SPI\_RXCTL to configure SPI Master Receive mode, and/or write to SPI\_TXCTL to configure SPI Master Transmit mode.

5. Finally, write to `SPI_RXCTL.REN` to enable the receive channel, and/or write to `SPI_TXCTL.TEN` to enable the transmit channel.
6. If `SPI_RXCTL.RTI` is enabled, a receive transfer is initiated upon enabling `SPI_CTL.EN`. If the Receive Word Counter is enabled (`SPI_RXCTL.RWCEN = 1`), then `SPI_RWC` must be non-zero for a transfer to initiate.

*ADDITIONAL INFORMATION:* If enabling both RX and TX DMA channels, but not enabling `SPI_TXCTL.TTI`, write to `SPI_RXCTL` after writing `SPI_CTL` and `SPI_TXCTL`, so that a transmit underrun can be prevented for the first transfer. Subsequent transfers are initiated as the SPI reads data from the RX shift register and writes to the SPI Receive FIFO. The SPI then requests a DMA write to memory. Upon a DMA grant, the DMA engine reads a word from the SPI Receive FIFO and writes to memory. New requests continue to be initiated as long as the receive FIFO does not fill up, provided that `SPI_RWC` does not become zero while `SPI_RXCTL.RWCEN = 1`.

7. If `SPI_TXCTL.TTI` is enabled, the SPI controller requests DMA reads from memory as long as there is space for more data in the transmit pipe. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO. As long as transmit data is available in the FIFO, and the `SPI_TWC` register is non-zero if `SPI_TXCTL.TWCEN = 1`, the SPI continues to initiate transfers until disabled.
8. If both `SPI_TXCTL.TTI` and `SPI_RXCTL.RTI` are enabled, the SPI controller requests a DMA read from memory as long as there is space for more data in the transmit pipe and the number of words written into the SPI is less than `SPI_TWC` if `SPI_TXCTL.TWCEN = 1`. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.

*ADDITIONAL INFORMATION:* As the SPI writes data from the transmit FIFO into the transmit shift Register, it initiates a transfer on the SPI link.

- a) Data received from the transfer is moved from the SPI receive shift register to the receive FIFO.
- b) The SPI controller requests a DMA write to memory.
- c) Upon a DMA grant, the DMA engine reads a word from the receive FIFO and writes to memory. Transfer will continue to be initiated as long as both receives and transmits can accommodate new data
9. If the receive pipe fills up due to unavailability of DMA grants, the transmit pipe will stall until the pipe is drained. If the transmit pipe fills up, the SPI will stop requesting for DMA writes. If the value in `SPI_RWC` expires, further write requests to DMA will stop. However, data already written into the transmit FIFO will transmit, and read requests to DMA will continue until the receive data is read from the receive FIFO.
10. The SPI then generates the programmed clock pulses on `SPI_CLK` and simultaneously shifts data out of `SPI_MOSI` while shifting data in from `SPI_MISO`. For receive transfers, the value in the shift register is loaded into the `SPI_RFIFO` register at the end of the transfer. For transmit transfers, the value in the `SPI_TFIFO` register is loaded into the shift register at the start of the transfer.

## Configuring SPI DMA Slave Mode Operation

When enabled as a slave with the DMA engine configured to transmit or receive data, the start of a transfer is triggered by a transition of the `SPI_SS` signal to the active-low state or by the first active edge of `SPI_CLK`, depending on the state of `SPI_CTL.CPHA`. The following steps illustrate the SPI receive or transmit DMA sequence in an SPI slave (in response to a master command). The SPI supports a receive DMA channel and a transmit DMA channel.

1. Write to the appropriate DMA registers to enable the SPI DMA channel and configure the necessary work units, access direction, word count, and so on.
2. Write to the `SPI_CTL`, `SPI_RXCTL` and `SPI_TXCTL` registers to define the mode of the serial link to be the same as the mode configured in the SPI master.
3. If the receive channel is enabled (`SPI_RXCTL.REN` is asserted), the following actions occur:
  - a. Once the slave select input is active, the slave starts receiving and transmitting data on active `SPI_CLK` edges.
  - b. The value in the shift register is loaded into the `SPI_RFIFO` register at the end of the transfer.
  - c. Once `SPI_RFIFO` has valid data, it requests a DMA write to memory.
  - d. Upon a DMA grant, the DMA engine reads a word from the receive FIFO and writes to memory.
  - e. As long as there is data in the receive FIFO, the SPI slave continues to request a DMA write to memory. The DMA engine continues to read a word from the FIFO and writes to memory until the `SPI_RWC` counts to zero. The SPI slave continues receiving words on active `SPI_CLK` edges as long as the `SPI_SS` input is active.
  - f. If the data collected in the receive pipe breaches the level set according to the `SPI_CTL.FCWM` field, and the DMA engine is unable to keep up with the receive rate, the slave may de-assert the `SPI_RDY` signal, throttling the master. The signal is deasserted as the DMA drains the receive FIFO. Alternatively, the `SPI_RXCTL.RDO` bit can decide if the incoming data is discarded or overwritten into the receive FIFO (when `SPI_CTL.FCEN` is inactive).
4. If the transmit channel is enabled (`SPI_TXCTL.TEN` is asserted), the following actions occur:
  - a. The SPI requests a DMA read from memory.
  - b. Upon a DMA grant, the DMA engine reads a word from memory and writes to the transmit FIFO.
  - c. The SPI then reads DMA data from the transmit FIFO and writes to the transmit shift register, awaiting the start of the next transfer.
  - d. Once the slave select input is active, the slave starts receiving and transmitting data on active `SPI_CLK` edges.
  - e. As long as there is room in the transmit FIFO, the SPI slave continues to request a DMA read from memory. The DMA engine continues to read a word from memory and write to the transmit FIFO.

until the SPI\_TWC register value counts down to 0. The SPI slave continues transmitting words on active SPI\_CLK edges as long as the  $\overline{\text{SPI\_SS}}$  input is active.

- f. If the number of outstanding data entries waiting for transmission in the transmit pipe breaches the level set according to the SPI\_CTL.FCWM field and the DMA is unable to keep up with the transmit rate, the slave may deassert the SPI\_RDY signal, throttling the master. The signal is deasserted as the DMA fills the transmit FIFO. Alternately the SPI\_TXCTL.TDU bit decides the state of the transmit data (when SPI\_CTL.FCEN is deasserted).
5. If both receive and transmit channels are enabled, the following actions occur after the actions stated above for each channel. Transfers will continue as long as both receives and transmits can accommodate new data.
    - a. If the receive pipe fills up due to unavailability of DMA grant, the SPI interface will stall the master by asserting the SPI\_RDY pin. This signal is deasserted as the DMA drains the receive FIFO. Alternately, the SPI\_RXCTL.RDO bit decides if the incoming data is discarded or overwritten into the receive FIFO (when SPI\_CTL.FCEN is deasserted).
    - b. If the transmit pipe fills up, the SPI will stop requesting for DMA writes until the pipe clears.
    - c. If there is an underflow problem in the transmit pipe, the slave will stall the master by deasserting SPI\_RDY while DMA fills the transmit FIFO. Alternately, the SPI\_TXCTL.TDU bit decides the state of the transmit data (when SPI\_CTL.FCEN is deasserted).

## SPI Register Descriptions

Serial Peripheral Interface (SPI) contains the following registers.

**Table 25-7:** SPI Module Registers

| Name       | Description                         |
|------------|-------------------------------------|
| SPI_CTL    | Control Register                    |
| SPI_RXCTL  | Receive Control Register            |
| SPI_TXCTL  | Transmit Control Register           |
| SPI_CLK    | Clock Rate Register                 |
| SPI_DLY    | Delay Register                      |
| SPI_SLVSEL | Slave Select Register               |
| SPI_RWC    | Received Word Count Register        |
| SPI_RWCR   | Received Word Count Reload Register |

**Table 25-7:** SPI Module Registers (Continued)

| Name         | Description                            |
|--------------|----------------------------------------|
| SPI_TWC      | Transmitted Word Count Register        |
| SPI_TWCR     | Transmitted Word Count Reload Register |
| SPI_IMSK     | Interrupt Mask Register                |
| SPI_IMSK_CLR | Interrupt Mask Clear Register          |
| SPI_IMSK_SET | Interrupt Mask Set Register            |
| SPI_STAT     | Status Register                        |
| SPI_ILAT     | Masked Interrupt Condition Register    |
| SPI_ILAT_CLR | Masked Interrupt Clear Register        |
| SPI_RFIFO    | Receive FIFO Data Register             |
| SPI_TFIFO    | Transmit FIFO Data Register            |

## Control Register

The SPI\_CTL register enables the SPI and configures settings for operating modes, communication protocols, and buffer operations.



### SPI\_CTL: Control Register - R/W

Reset = 0x0000 0050

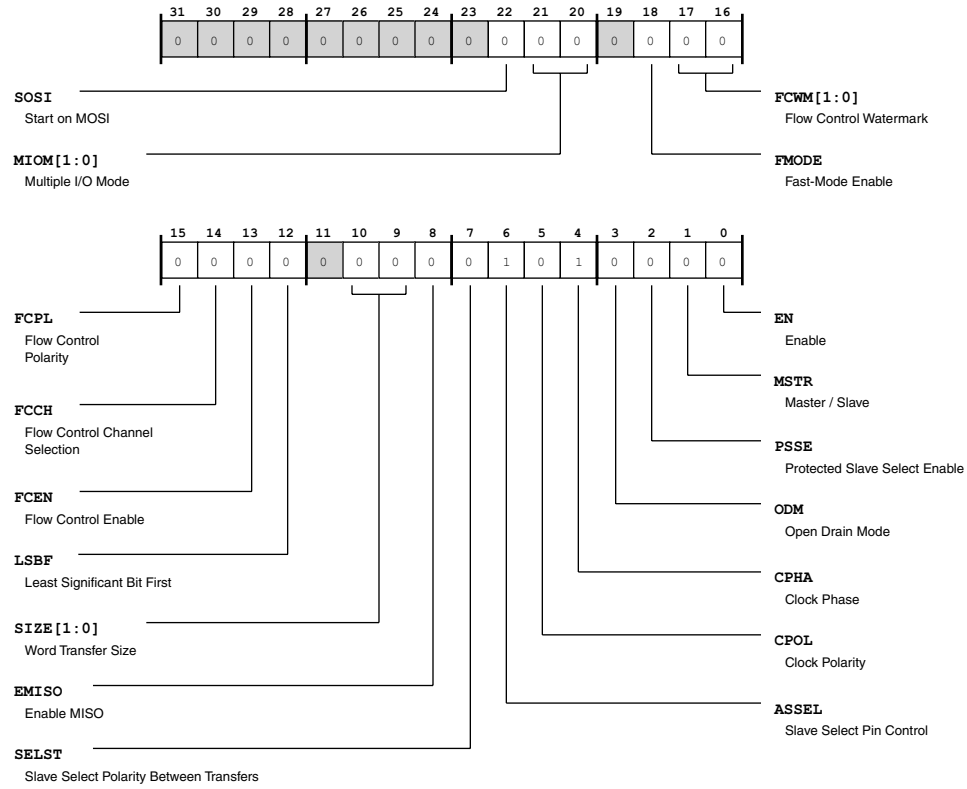


**Figure 25-13:** SPI\_CTL Register Diagram

**Table 25-8:** SPI\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | SOSI     | <p>Start on MOSI.</p> <p>The SPI_CTL.SOSI bit is valid only when SPI_CTL.MIOM is enabled for either DIOM or QIOM, and this bit selects the starting pin and the bit placement on pins for these modes.</p> <p>In DIOM, enabling SPI_CTL.SOSI sends bit 1 on the SPI_MOSI pin and sends bit 2 on the SPI_MISO pin. Disabling SPI_CTL.SOSI reverses this order.</p> <p>In QIOM, enabling SPI_CTL.SOSI sends bit 1 on the SPI_MOSI pin, sends bit 2 on the SPI_MISO pin, sends bit 3 on SPI_D2 pin, and sends bit 4 on the SPI_D3 pin. Disabling SPI_CTL.SOSI reverses this order.</p> |
|         |          | 0 Bit 1 on MISO (DIOM) or on D3 (QIOM)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Bit 1 on MOSI (DIOM and QIOM)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 21:20   | MIOM     | <p>Multiple I/O Mode.</p> <p>The SPI_CTL.MIOM bits enable SPI operation in dual I/O mode (DIOM) or quad I/O mode (QIOM).</p> <p>These bits may only be changed when the SPI is disabled (SPI_CTL.EN =0).</p>                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 0 No MIOM (disabled)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 DIOM operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 2 QIOM operation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 3 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 18      | FMODE    | <p>Fast-Mode Enable.</p> <p>The SPI_CTL.FMODE bit enables fast mode operation for SPI receive transfers. SPI transmit operations in fast mode are the same as normal mode.</p>                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 25-8:** SPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17:16   | FCWM     | Flow Control Watermark.<br>The SPI_CTL.FCWM bits select the watermark level of the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer) that triggers flow control operation. These bits are applicable only when the SPI is a slave (SPI_CTL.MSTR = 0) and flow control is enabled (SPI_CTL.FCEN = 1). When the watermark condition is met, the SPI slave de-asserts the SPI_RDY pin. |
|         |          | 0      TFIFO empty or RFIFO full                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      TFIFO 75% or more empty, or RFIFO full                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 2      TFIFO 50% or more empty, or RFIFO full                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 3      Reserved                                                                                                                                                                                                                                                                                                                                                                                             |
| 15      | FCPL     | Flow Control Polarity.<br>The SPI_CTL.FCPL bit selects flow control polarity for the SPI_RDY pin when flow control is enabled. When the SPI_RDY pin is active, the SPI is indicating it's ready for data transfer.                                                                                                                                                                                          |
|         |          | 0      Active-low RDY                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Active-high RDY                                                                                                                                                                                                                                                                                                                                                                                      |
| 14      | FCCH     | Flow Control Channel Selection.<br>The SPI_CTL.FCCH bit selects whether the SPI applies flow control to the transmit channel (SPI_TFIFO buffer) or receive channel (SPI_RFIFO buffer). This bit is applicable only when the SPI is a slave and flow control is enabled.                                                                                                                                     |
|         |          | 0      Flow control on RX buffer                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1      Flow control on TX buffer                                                                                                                                                                                                                                                                                                                                                                            |

**Table 25-8:** SPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | FCEN     | Flow Control Enable.<br>The SPI_CTL.FCEN bit enables SPI flow control operation, which permits slow slave devices to interface with fast master devices. This bit controls operation of the SPI_RDY pin. Note that options for flow control operation are available using the SPI_CTL.FCCH, SPI_CTL.FCPL, and SPI_CTL.FCWM bits. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                    |
| 12      | LSBF     | Least Significant Bit First.<br>The SPI_CTL.LSBF bit selects whether the SPI transmits/receives data as LSB first (little endian) or MSB first (big endian). This bit may only be changed when the SPI is disabled.                                                                                                              |
|         |          | 0      MSB sent/received first (big endian)                                                                                                                                                                                                                                                                                      |
|         |          | 1      LSB sent/received first (little endian)                                                                                                                                                                                                                                                                                   |
| 10:9    | SIZE     | Word Transfer Size.<br>The SPI_CTL.SIZE bits select the SPI transfer word size as 8, 16 or 32 bits. To ensure correct operation, both the master and slave must be configured with the same word size. This bit may only be changed when the SPI is disabled (SPI_CTL.EN = 0).                                                   |
|         |          | 0      8-bit word                                                                                                                                                                                                                                                                                                                |
|         |          | 1      16-bit word                                                                                                                                                                                                                                                                                                               |
|         |          | 2      32-bit word                                                                                                                                                                                                                                                                                                               |
|         |          | 3      Reserved                                                                                                                                                                                                                                                                                                                  |
| 8       | EMISO    | Enable MISO.<br>The SPI_CTL.EMISO bit enables master-in-slave-out (MISO) mode. This SPI mode is applicable only when the SPI is a slave.                                                                                                                                                                                         |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                    |

**Table 25-8:** SPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | SELST    | Slave Select Polarity Between Transfers.<br>The SPI_CTL.SELST bit selects the state (polarity) for the $\overline{\text{SPI\_SELN}}$ pin in-between SPI transfers when the SPI is a master and slave select assertion is enabled (SPI_CTL.ASSEL =1).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 0 De-assert slave select (high)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Assert slave select (low)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 6       | ASSEL    | Slave Select Pin Control.<br>The SPI_CTL.ASSEL bit selects whether the SPI hardware sets the $\overline{\text{SPI\_SELN}}$ pin output value (ignoring the slave select SPI_SLVSEL.SSEL1 - SPI_SLVSEL.SSEL7 bits) or whether software control of the slave select bits set the $\overline{\text{SPI\_SELN}}$ pin output value. This feature is applicable only when the SPI is a master.<br>When hardware control is enabled, the $\overline{\text{SPI\_SELN}}$ pin output is asserted during the transfers, and the pin polarity between transfers is selected by the SPI_CTL.SELST bit.<br>When software control is enabled, the $\overline{\text{SPI\_SELN}}$ pin output value is set through software control of the slave select bits, and as such, the pin may either remain asserted (low) or be deasserted between transfers. |
|         |          | 0 Software Slave Select Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Hardware Slave Select Control                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 5       | CPOL     | Clock Polarity.<br>The SPI_CTL.CPOL bit selects whether the SPI uses an active-low or active-high signal for the SPI clock (SPI_CLK). This bit works with the SPI_CTL.CPHA bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit may only be changed when the SPI is disabled.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 0 Active-high SPI CLK                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1 Active-low SPI CLK                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

**Table 25-8:** SPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 4       | CPHA     | Clock Phase.<br>The SPI_CTL.CPHA bit selects whether the SPI starts toggling the signal for the SPI clock (SPI_CLK) from the start of the first data bit or from the middle of the first data bit. The SPI_CTL.CPHA bit works with the SPI_CTL.CPOL bit to select combinations of clock phase and polarity for the SPI_CLK pin. This bit may only be changed when the SPI is disabled.                                                                                                                                                                                                                                                                                                                                                                    |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | SPI CLK toggles from middle |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | SPI CLK toggles from start  |
| 3       | ODM      | Open Drain Mode.<br>The SPI_CTL.ODM bit configures the data output pins (SPI_MOSI and SPI_MISO) to behave as open drain outputs, which prevents contention and possible damage to pin drivers in multi-master or multi-slave SPI systems.<br>When SPI_CTL.ODM is enabled and the SPI is a master, the SPI three-states the SPI_MOSI pin when the data driven out on MOSI is a logic-high. The SPI does not three-state the SPI_MOSI pin when the driven data is a logic-low.<br>When SPI_CTL.ODM is enabled and the SPI is a slave, the SPI three-states the SPI_MISO pin when the data driven out on SPI_MISO is a logic-high.<br>Note that an external pull-up resistor is required on both the SPI_MOSI and SPI_MISO pins when SPI_CTL.ODM is enabled. |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Disable                     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Enable                      |
| 2       | PSSE     | Protected Slave Select Enable.<br>The SPI_CTL.PSSE bit enables the $\overline{\text{SPI\_SS}}$ pin to provide error detection input in a multi-master environment when the SPI is in master mode. If some other device in the system asserts the $\overline{\text{SPI\_SS}}$ pin while SPI is enabled as master (and SPI_CTL.PSSE is enabled), this condition causes a mode fault error.                                                                                                                                                                                                                                                                                                                                                                  |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Disable                     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Enable                      |

**Table 25-8:** SPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | MSTR     | Master / Slave.<br>The SPI_CTL.MSTR bit toggles the SPI between master mode and slave mode. This bit may only be changed when the SPI is disabled. |
|         |          | 0 Slave                                                                                                                                            |
|         |          | 1 Master                                                                                                                                           |
| 0       | EN       | Enable.<br>The SPI_CTL.EN bit enables SPI operation.                                                                                               |
|         |          | 0 Disable SPI module                                                                                                                               |
|         |          | 1 Enable                                                                                                                                           |

## Receive Control Register

The SPI\_RXCTL register enables the SPI receive channel, initiates receive transfers, and configures SPI\_RFIFO buffer watermark settings.

### SPI\_RXCTL: Receive Control Register - R/W

Reset = 0x0000 0000

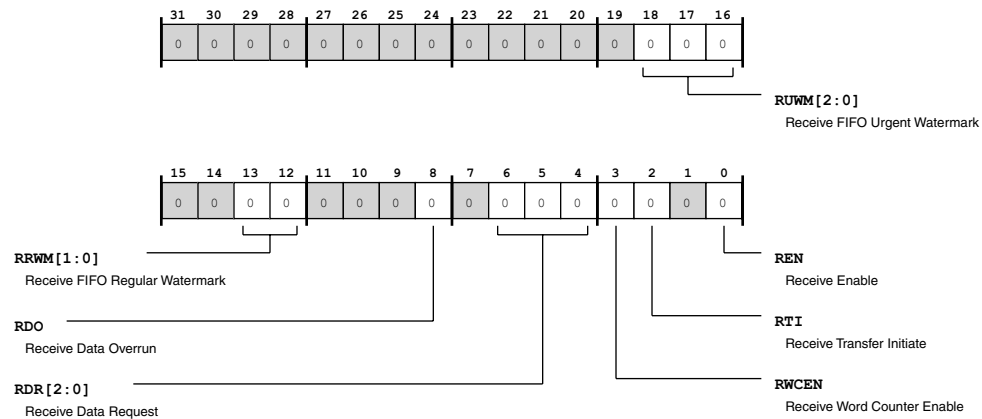


**Figure 25-14:** SPI\_RXCTL Register Diagram

**Table 25-9:** SPI\_RXCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | RUWM     | Receive FIFO Urgent Watermark.<br>The SPI_RXCTL.RUWM bits select the receive FIFO (SPI_RFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.RUWM interrupt. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the de-assertion condition for any SPI_ILAT.RUWM interrupts that are latched. |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 25% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2 50% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 3 75% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 4 Full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 5 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 6 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 13:12   | RRWM     | Receive FIFO Regular Watermark.<br>The SPI_RXCTL.RRWM bits select the receive FIFO (SPI_RFIFO) watermark level for regular data bus requests. When an urgent SPI_RFIFO watermark is enabled with SPI_RXCTL.RUWM, the SPI_RXCTL.RRWM selection is used as the de-assertion condition for any SPI_ILAT.RUWM interrupts that are latched.                                                                                     |
|         |          | 0 Empty RFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 25% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2 50% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 3 75% full RFIFO                                                                                                                                                                                                                                                                                                                                                                                                           |



**Table 25-9:** SPI\_RXCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | RDO      | Receive Data Overrun.<br>The SPI_RXCTL.RDO bit selects handling for receive data requests when the receive buffer (SPI_RFIFO) is full. If enabled and SPI_RFIFO is full, the SPI overwrites old data in the buffer with incoming data. If disabled and SPI_RFIFO is full, the SPI keeps old data in the buffer and discards incoming data. |
|         |          | 0 KeDiscard incoming data if SPI_RFIFO is full                                                                                                                                                                                                                                                                                             |
|         |          | 1 Overwrite old data if SPI_RFIFO is full                                                                                                                                                                                                                                                                                                  |
| 6:4     | RDR      | Receive Data Request.<br>The SPI_RXCTL.RDR bits select receive FIFO (SPI_RFIFO) watermark conditions that direct the SPI to generate a receive status interrupt.                                                                                                                                                                           |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Not empty RFIFO                                                                                                                                                                                                                                                                                                                          |
|         |          | 2 25% full RFIFO                                                                                                                                                                                                                                                                                                                           |
|         |          | 3 50% full RFIFO                                                                                                                                                                                                                                                                                                                           |
|         |          | 4 75% full RFIFO                                                                                                                                                                                                                                                                                                                           |
|         |          | 5 Full RFIFO                                                                                                                                                                                                                                                                                                                               |
|         |          | 6 Reserved                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 Reserved                                                                                                                                                                                                                                                                                                                                 |
| 3       | RWCEN    | Receive Word Counter Enable.<br>The SPI_RXCTL.RWCEN bit enables the decrement of the SPI_RWC register when the count is not zero and SPI_RXCTL.RTI is enabled. Enabling SPI_RXCTL.RWCEN prevents receive overrun errors from occurring. The SPI_RXCTL.RWCEN bit is valid only when the SPI is a master.                                    |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                   |

Table 25-9: SPI\_RXCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | RTI      | Receive Transfer Initiate.<br>The SPI_RXCTL.RTI bit enables initiation of receive transfers if the receive FIFO (SPI_RFIFO) is not full. The bit also enables this initiation if SPI_RWC is not zero when SPI_RXCTL.RWCEN is enabled. Enabling SPI_RXCTL.RTI prevents receive overrun errors from occurring. The SPI_RXCTL.RTI bit is valid only when the SPI is a master. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                   |
| 0       | REN      | Receive Enable.<br>The SPI_RXCTL.REN bit enables SPI receive channel operation.                                                                                                                                                                                                                                                                                            |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                   |

Transmit Control Register

The SPI\_TXCTL register enables the SPI transmit channel, initiates transmit transfers, and configures SPI\_TFIFO buffer watermark settings.

SPI\_TXCTL: Transmit Control Register - R/W

Reset = 0x0000 0000

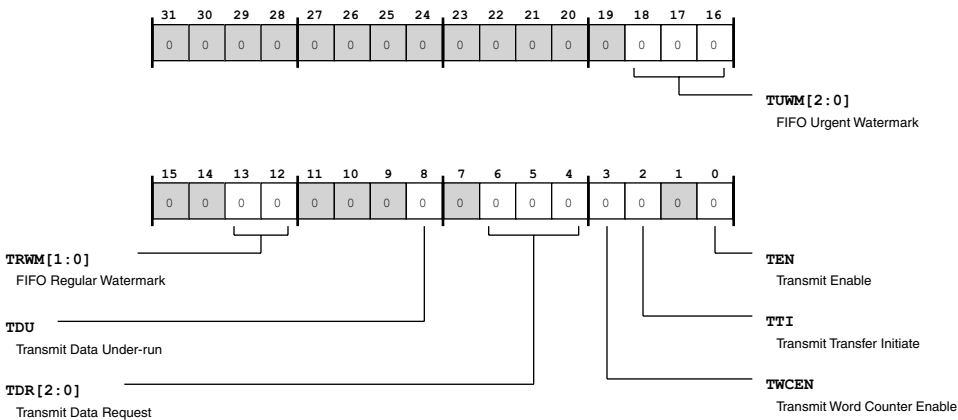


Figure 25-15: SPI\_TXCTL Register Diagram

**Table 25-10:** SPI\_TXCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | TUWM     | <p>FIFO Urgent Watermark.</p> <p>The SPI_TXCTL.TUWM bits select the transmit FIFO (SPI_TFIFO) watermark level for urgent data bus requests. The SPI also uses this watermark level for generation of the SPI_ILAT.TUWM interrupt. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the de-assertion condition for any SPI_ILAT.TUWM interrupts that are latched.</p> |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 25% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 2 50% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 3 75% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 4 Empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 13:12   | TRWM     | <p>FIFO Regular Watermark.</p> <p>The SPI_TXCTL.TRWM bits select the transmit FIFO (SPI_TFIFO) watermark level for regular data bus requests. When an urgent SPI_TFIFO watermark is enabled with SPI_TXCTL.TUWM, the SPI_TXCTL.TRWM selection is used as the de-assertion condition for any SPI_ILAT.TUWM interrupts that are latched.</p>                                                                                     |
|         |          | 0 Full TFIFO                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 25% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 2 50% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 3 75% empty TFIFO                                                                                                                                                                                                                                                                                                                                                                                                              |

**Table 25-10:** SPI\_TXCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | TDU      | Transmit Data Under-run.<br>The SPI_TXCTL.TDU bit selects handling for transmit data requests when the transmit buffer (SPI_TFIFO) is empty. If enabled and SPI_TFIFO is empty, the SPI transmits zero as data. If disabled and SPI_TFIFO is empty, the SPI transmits the last word in the buffer as data.                        |
|         |          | 0 Send last word when SPI_TFIFO is empty                                                                                                                                                                                                                                                                                          |
|         |          | 1 Send zeros when SPI_TFIFO is empty                                                                                                                                                                                                                                                                                              |
| 6:4     | TDR      | Transmit Data Request.<br>The SPI_TXCTL.TDR bits select transmit FIFO (SPI_TFIFO) watermark conditions that direct the SPI to generate a transmit status interrupt.                                                                                                                                                               |
|         |          | 0 Disabled                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 Not full TFIFO                                                                                                                                                                                                                                                                                                                  |
|         |          | 2 25% empty TFIFO                                                                                                                                                                                                                                                                                                                 |
|         |          | 3 50% empty TFIFO                                                                                                                                                                                                                                                                                                                 |
|         |          | 4 75% empty TFIFO                                                                                                                                                                                                                                                                                                                 |
|         |          | 5 Empty TFIFO                                                                                                                                                                                                                                                                                                                     |
| 3       | TWCEN    | Transmit Word Counter Enable.<br>The SPI_TXCTL.TWCEN bit enables the decrement of the transmit word count (SPI_TWC) register when the count is not zero and SPI_TXCTL.TTI is enabled. Enabling SPI_TXCTL.TWCEN prevents transmit under-run errors from occurring. The SPI_TXCTL.TWCEN bit is valid only when the SPI is a master. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                          |

**Table 25-10:** SPI\_TXCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | TTI      | Transmit Transfer Initiate.<br>The SPI_TXCTL.TTI bit enables initiation of transmit transfers if the transmit FIFO (SPI_TFIFO) is not empty. The bit also enables this initiation if SPI_TWC is not zero when SPI_TXCTL.TWCEN is enabled. Enabling SPI_TXCTL.TTI prevents transmit underrun errors from occurring. The SPI_TXCTL.TTI bit is valid only when the SPI is a master. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                    |
| 0       | TEN      | Transmit Enable.<br>The SPI_TXCTL.TEN bit enables SPI transmit channel operation.                                                                                                                                                                                                                                                                                                |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                    |

## Clock Rate Register

The SPI\_CLK register selects the baud rate for SPI data transfers, relating this rate to the SPI serial clock (SCK) and the system clock (SCLK).

**SPI\_CLK: Clock Rate Register - R/W**

Reset = 0x0000 0000

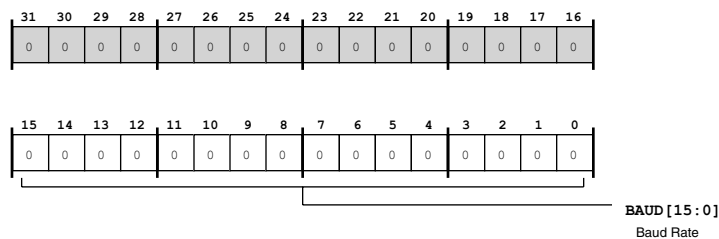


**Figure 25-16:** SPI\_CLK Register Diagram

Table 25-11: SPI\_CLK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | BAUD     | Baud Rate.<br>The SPI_CLK.BAUD bits set the SPI baud rate according to the formula:<br>$BAUD = (SCLK / \text{SPI Clock}) - 1$ |

Delay Register

The SPI\_DLY register selects a transfer delay and the lead/lag timing between slave select signals and SPI clock edge assertion/de-assertion.

SPI\_DLY: Delay Register - R/W

Reset = 0x0000 0301

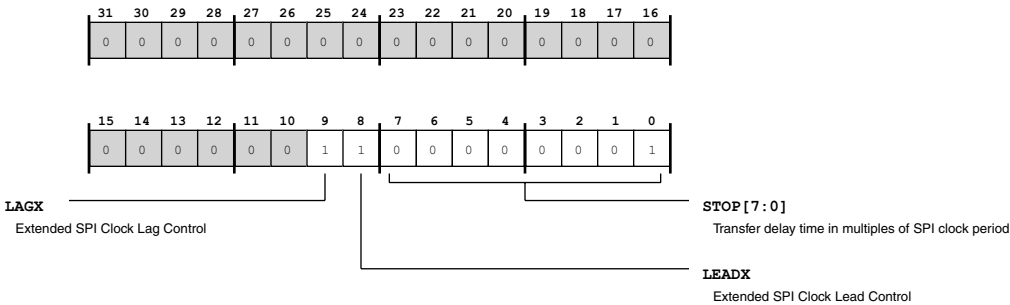


Figure 25-17: SPI\_DLY Register Diagram

Table 25-12: SPI\_DLY Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | LAGX     | Extended SPI Clock Lag Control.<br>The SPI_DLY.LAGX bit enables insertion of a 1-SPI_CLK cycle lag (extend lag) in the timing between the slave select ( <u>SPI_SELN</u> ) assertion and first SPI Clock edge. |
|         |          | 0 Disable                                                                                                                                                                                                      |
|         |          | 1 Enable                                                                                                                                                                                                       |

**Table 25-12:** SPI\_DLY Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | LEADX    | Extended SPI Clock Lead Control.<br>The SPI_DLY.LEADX bit enables insertion of a 1-SPI_CLK cycle lead (extend lead) in the timing between the slave select ( $\overline{\text{SPI\_SELN}}$ ) de-assertion and last SPI Clock edge.                                                                                                                                                                                          |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                               |
| 7:0     | STOP     | Transfer delay time in multiples of SPI clock period.<br>The SPI_DLY.STOP bits select a delay (number of stop bits in multiples of SPI Clock duration) at the end of each SPI transfer. The default delay is the minimum value required to comply with the SPI protocol (1-bit duration). The SPI_DLY.STOP bits may be programmed with smaller delay values, resulting in continuous operation (for example, stop bits =0). |

## Slave Select Register

The SPI\_SLVSEL register enables the  $\overline{\text{SPI\_SELN}}$  pins for input and indicates the state (high or low) of these pins when enabled.

SPI\_SLVSEL: Slave Select Register - R/W

Reset = 0x0000 fe00

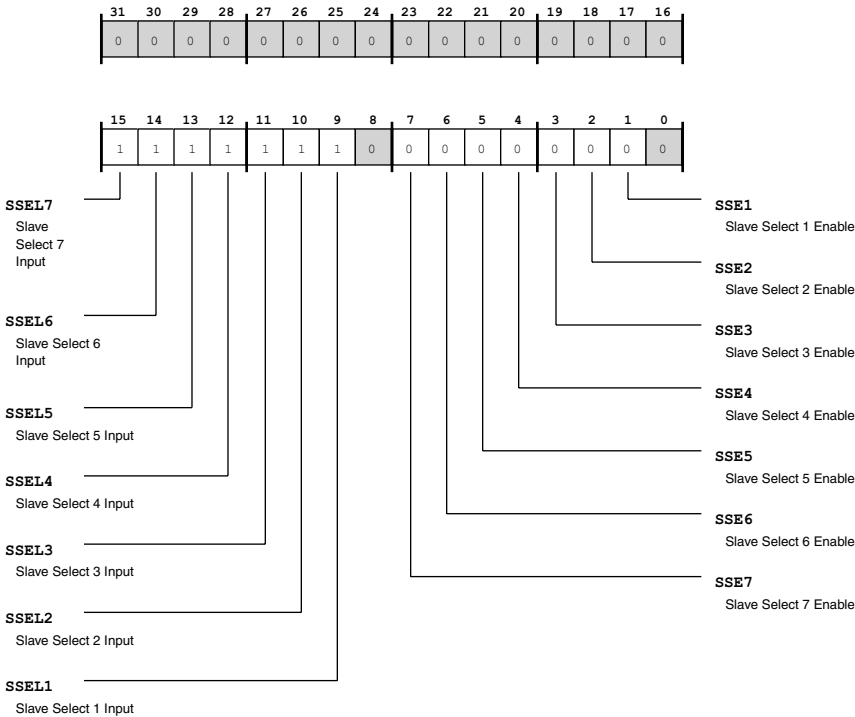


Figure 25-18: SPI\_SLVSEL Register Diagram

Table 25-13: SPI\_SLVSEL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                  |      |
|---------|----------|----------------------------------------------------------------------------------------------------------|------|
| 15      | SSEL7    | Slave Select 7 Input.<br>The SPI_SLVSEL.SSEL7 bit state indicates the value on the related SPI_SELN pin. |      |
|         |          | 0                                                                                                        | Low  |
|         |          | 1                                                                                                        | High |
| 14      | SSEL6    | Slave Select 6 Input.<br>The SPI_SLVSEL.SSEL6 bit state indicates the value on the related SPI_SELN pin. |      |
|         |          | 0                                                                                                        | Low  |
|         |          | 1                                                                                                        | High |



**Table 25-13:** SPI\_SLVSEL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 13      | SSEL5    | Slave Select 5 Input.<br>The SPI_SLVSEL.SSEL5 bit state indicates the value on the related $\overline{\text{SPI\_SELN}}$ pin. |
|         |          | 0 Low                                                                                                                         |
|         |          | 1 High                                                                                                                        |
| 12      | SSEL4    | Slave Select 4 Input.<br>The SPI_SLVSEL.SSEL4 bit state indicates the value on the related $\overline{\text{SPI\_SELN}}$ pin. |
|         |          | 0 Low                                                                                                                         |
|         |          | 1 High                                                                                                                        |
| 11      | SSEL3    | Slave Select 3 Input.<br>The SPI_SLVSEL.SSEL3 bit state indicates the value on the related $\overline{\text{SPI\_SELN}}$ pin. |
|         |          | 0 Low                                                                                                                         |
|         |          | 1 High                                                                                                                        |
| 10      | SSEL2    | Slave Select 2 Input.<br>The SPI_SLVSEL.SSEL2 bit state indicates the value on the related $\overline{\text{SPI\_SELN}}$ pin. |
|         |          | 0 Low                                                                                                                         |
|         |          | 1 High                                                                                                                        |
| 9       | SSEL1    | Slave Select 1 Input.<br>The SPI_SLVSEL.SSEL1 bit state indicates the value on the related $\overline{\text{SPI\_SELN}}$ pin. |
|         |          | 0 Low                                                                                                                         |
|         |          | 1 High                                                                                                                        |

**Table 25-13:** SPI\_SLVSEL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | SSE7     | Slave Select 7 Enable.<br>The SPI_SLVSEL.SSE7 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. If disabled, the SPI three-states the related $\overline{\text{SPI\_SELN}}$ pin. When the SPI is a slave, the master selecting the slave asserts the input during the transfer. The input may be de-asserted or remain asserted between transfers. While the input is de-asserted, the SPI ignores SPI Clock, ignores inputs, and three-states outputs. When the SPI is a master and the SPI_CTL.PSSE bit is enabled, the input serves as an error detection input in a multi-master environment. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 6       | SSE6     | Slave Select 6 Enable.<br>The SPI_SLVSEL.SSE6 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 5       | SSE5     | Slave Select 5 Enable.<br>The SPI_SLVSEL.SSE5 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 4       | SSE4     | Slave Select 4 Enable.<br>The SPI_SLVSEL.SSE4 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

**Table 25-13:** SPI\_SLVSEL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | SSE3     | Slave Select 3 Enable.<br>The SPI_SLVSEL.SSE3 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information. |
|         |          | 0                      Disable                                                                                                                                                   |
|         |          | 1                      Enable                                                                                                                                                    |
| 2       | SSE2     | Slave Select 2 Enable.<br>The SPI_SLVSEL.SSE2 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information. |
|         |          | 0                      Disable                                                                                                                                                   |
|         |          | 1                      Enable                                                                                                                                                    |
| 1       | SSE1     | Slave Select 1 Enable.<br>The SPI_SLVSEL.SSE1 bit enables the related $\overline{\text{SPI\_SELN}}$ pin for input. See the SPI_SLVSEL.SSE7 bit description for more information. |
|         |          | 0                      Disable                                                                                                                                                   |
|         |          | 1                      Enable                                                                                                                                                    |

## Received Word Count Register

The SPI\_RWC register holds a count of the number of words remaining to be received by the SPI. To start the decrement of the word count in SPI\_RWC, enable the receive word counter (SPI\_RXCTL.RWCEN = 1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the receive finish interrupt (SPI\_ILAT.RF). In DMA mode, the SPI uses the SPI\_RWC to ensure that the number of frames received during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI\_RWC registers should match the word count in the DMA configuration. The SPI\_RWC maintains the number of frames to be received in a transfer. The SPI\_RWC should only be changed when the counter is disabled.

SPI\_RWC: Received Word Count Register - R/W

Reset = 0x0000 0000

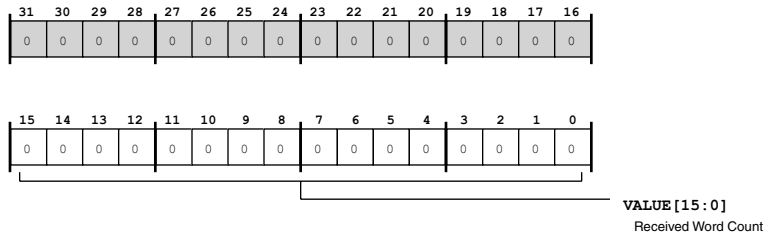


Figure 25-19: SPI\_RWC Register Diagram

Table 25-14: SPI\_RWC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                              |
|---------|----------|--------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Received Word Count.<br>The SPI_RWC.VALUE bits hold the receive transfer word count. |

Received Word Count Reload Register

The SPI\_RWCR register holds the receive word count value that the SPI loads into the SPI\_RWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI\_RWCR should only be changed when the counter is disabled.

SPI\_RWCR: Received Word Count Reload Register - R/W

Reset = 0x0000 0000

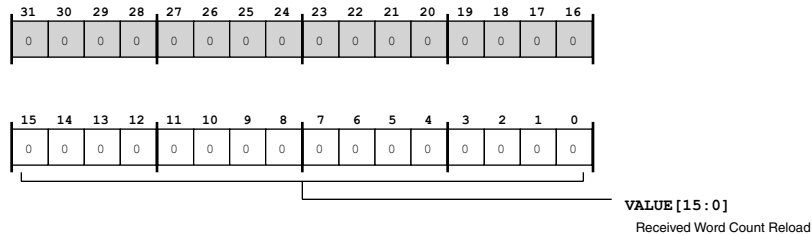


Figure 25-20: SPI\_RWCR Register Diagram

**Table 25-15:** SPI\_RWCR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Received Word Count Reload.<br>The SPI_RWCR.VALUE bits hold the receive transfer word count reload value. |

## Transmitted Word Count Register

The SPI\_TWC register holds a count of the number of words remaining to be transmitted by the SPI. To start the decrement of the word count in SPI\_TWC, enable the transmit word counter (SPI\_TXCTL.TWCEN =1). The SPI uses the word count to control the duration of transfers and to signal the completion of a burst of transfers with the transmit finish interrupt. In DMA mode, the SPI uses the SPI\_TWC to ensure that the number of frames transmitted during a DMA transfer is equal to the number of words programmed in the DMA channel controller. The values programmed into the SPI\_TWC registers should match the word count in the DMA configuration. The SPI\_TWC maintains the number of frames to be transmitted in a transfer. The SPI\_TWC should only be changed when the counter is disabled.

### SPI\_TWC: Transmitted Word Count Register - R/W

Reset = 0x0000 0000

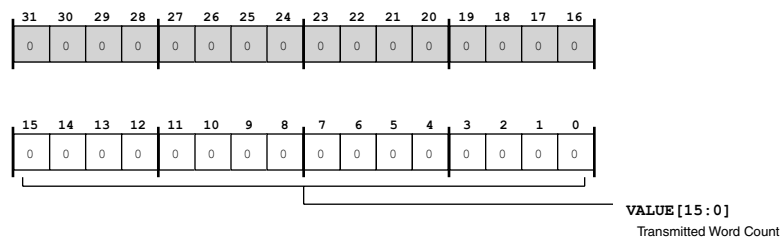


**Figure 25-21:** SPI\_TWC Register Diagram

**Table 25-16:** SPI\_TWC Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                  |
|---------|----------|------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Transmitted Word Count.<br>The SPI_TWC.VALUE bits hold the transmit transfer word count. |

## Transmitted Word Count Reload Register

The SPI\_TWCR register holds the transmit word count value that the SPI loads into the SPI\_TWC register when the transfer count decrements to zero. To prevent the SPI from reloading the counter, use zero for the reload count value. The SPI\_TWCR should only be changed when the counter is disabled.

SPI\_TWCR: Transmitted Word Count Reload Register - R/W

Reset = 0x0000 0000

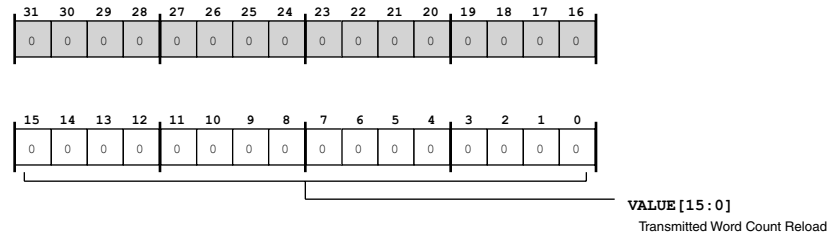


Figure 25-22: SPI\_TWCR Register Diagram

Table 25-17: SPI\_TWCR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Transmitted Word Count Reload.<br>The SPI_TWCR.VALUE bits hold the transmit transfer word count reload value. |

Interrupt Mask Register

The SPI\_IMSK register unmask (enables) or mask (disables) SPI interrupts. When a condition is indicated by a bit in the SPI\_STAT register and the corresponding interrupt is unmasked in SPI\_IMSK, the SPI latches the interrupt's bit in the SPI\_ILAT register, queuing the interrupt for service.

**SPI\_IMSK: Interrupt Mask Register - R/WE**

Reset = 0x0000 0000

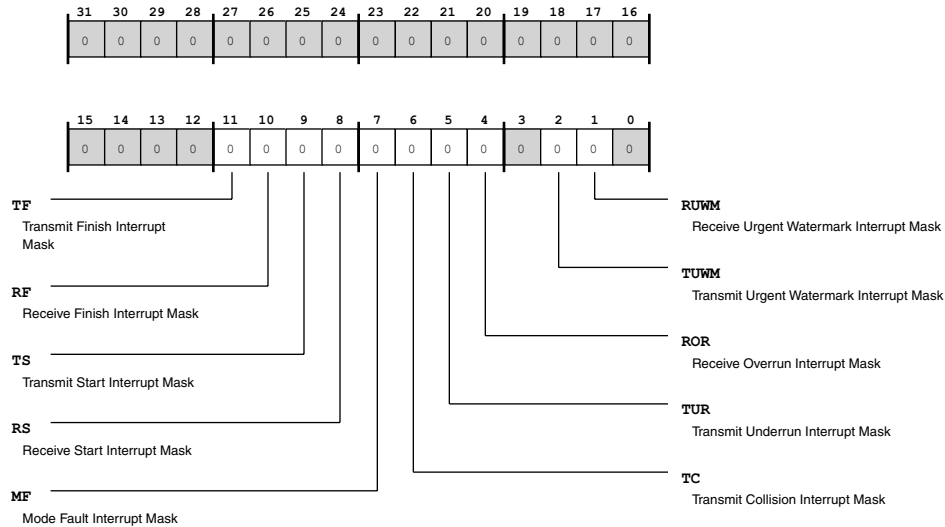


**Figure 25-23:** SPI\_IMSK Register Diagram

**Table 25-18:** SPI\_IMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 11      | TF       | Transmit Finish Interrupt Mask.  |
|         |          | 0      Disable (mask) interrupt  |
|         |          | 1      Enable (unmask) interrupt |
| 10      | RF       | Receive Finish Interrupt Mask.   |
|         |          | 0      Disable (mask) interrupt  |
|         |          | 1      Enable (unmask) interrupt |
| 9       | TS       | Transmit Start Interrupt Mask.   |
|         |          | 0      Disable (mask) interrupt  |
|         |          | 1      Enable (unmask) interrupt |

**Table 25-18:** SPI\_IMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                   |                           |
|---------|----------|-------------------------------------------|---------------------------|
| 8       | RS       | Receive Start Interrupt Mask.             |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 7       | MF       | Mode Fault Interrupt Mask.                |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 6       | TC       | Transmit Collision Interrupt Mask.        |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 5       | TUR      | Transmit Underrun Interrupt Mask.         |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 4       | ROR      | Receive Overrun Interrupt Mask.           |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 2       | TUWM     | Transmit Urgent Watermark Interrupt Mask. |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |
| 1       | RUWM     | Receive Urgent Watermark Interrupt Mask.  |                           |
|         |          | 0                                         | Disable (mask) interrupt  |
|         |          | 1                                         | Enable (unmask) interrupt |



## Interrupt Mask Clear Register

The SPI\_IMSK\_CLR register permits clearing individual mask bits in the SPI\_IMSK register without affecting other bits in the register. Use write-1-to-clear on a bit in SPI\_IMSK\_CLR to clear the corresponding bit in the SPI\_IMSK register.

### SPI\_IMSK\_CLR: Interrupt Mask Clear Register - R/W

Reset = 0x0000 0000

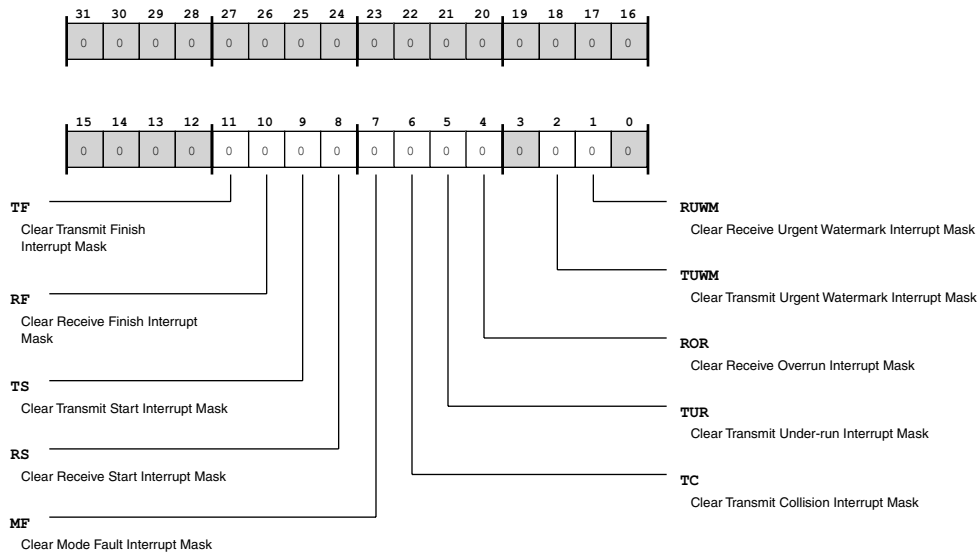


Figure 25-24: SPI\_IMSK\_CLR Register Diagram

Table 25-19: SPI\_IMSK\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                  |
|---------|----------|------------------------------------------|
| 11      | TF       | Clear Transmit Finish Interrupt Mask.    |
| 10      | RF       | Clear Receive Finish Interrupt Mask.     |
| 9       | TS       | Clear Transmit Start Interrupt Mask.     |
| 8       | RS       | Clear Receive Start Interrupt Mask.      |
| 7       | MF       | Clear Mode Fault Interrupt Mask.         |
| 6       | TC       | Clear Transmit Collision Interrupt Mask. |
| 5       | TUR      | Clear Transmit Under-run Interrupt Mask. |
| 4       | ROR      | Clear Receive Overrun Interrupt Mask.    |

Table 25-19: SPI\_IMSK\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                         |
|---------|----------|-------------------------------------------------|
| 2       | TUWM     | Clear Transmit Urgent Watermark Interrupt Mask. |
| 1       | RUWM     | Clear Receive Urgent Watermark Interrupt Mask.  |

Interrupt Mask Set Register

The SPI\_IMSK\_SET register permits setting individual mask bits in the SPI\_IMSK register without affecting other bits in the register. Use write-1-to-set on a bit in SPI\_IMSK\_SET to set the corresponding bit in the SPI\_IMSK register.

SPI\_IMSK\_SET: Interrupt Mask Set Register - R/W

Reset = 0x0000 0000

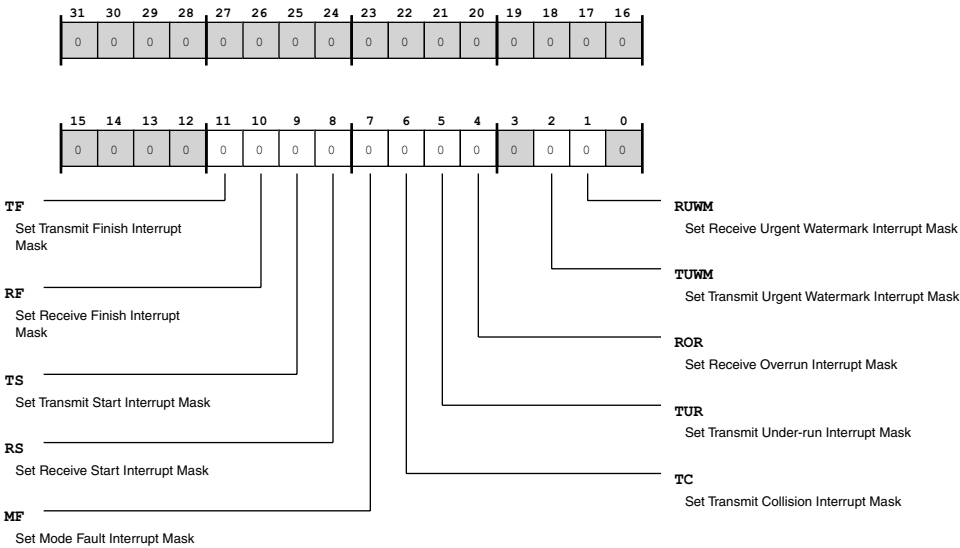


Figure 25-25: SPI\_IMSK\_SET Register Diagram

Table 25-20: SPI\_IMSK\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration             |
|---------|----------|-------------------------------------|
| 11      | TF       | Set Transmit Finish Interrupt Mask. |
| 10      | RF       | Set Receive Finish Interrupt Mask.  |
| 9       | TS       | Set Transmit Start Interrupt Mask.  |

**Table 25-20:** SPI\_IMSK\_SET Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                       |
|---------|----------|-----------------------------------------------|
| 8       | RS       | Set Receive Start Interrupt Mask.             |
| 7       | MF       | Set Mode Fault Interrupt Mask.                |
| 6       | TC       | Set Transmit Collision Interrupt Mask.        |
| 5       | TUR      | Set Transmit Under-run Interrupt Mask.        |
| 4       | ROR      | Set Receive Overrun Interrupt Mask.           |
| 2       | TUWM     | Set Transmit Urgent Watermark Interrupt Mask. |
| 1       | RUWM     | Set Receive Urgent Watermark Interrupt Mask.  |

## Status Register

The SPI\_STAT register indicates SPI status including FIFO status, error conditions, and interrupt conditions. When an interrupt condition from this register is unmasked (enabled) by the corresponding bit in the SPI\_IMSK register, the interrupt is latched into the corresponding bit in the SPI\_ILAT register.

SPI\_STAT: Status Register - R/W

Reset = 0x0140 0001

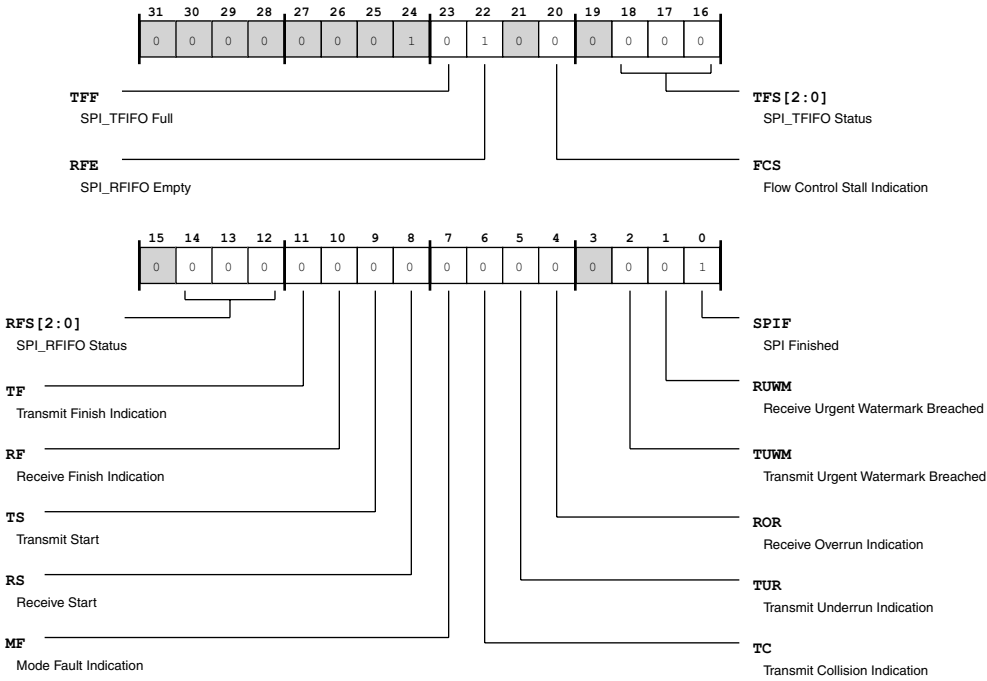


Figure 25-26: SPI\_STAT Register Diagram

Table 25-21: SPI\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                         |                   |
|---------|----------|-------------------------------------------------------------------------------------------------|-------------------|
| 23      | TFF      | SPI_TFIFO Full.<br>The SPI_STAT.TFF bit indicates whether the SPI_TFIFO is full or not full.    |                   |
|         |          | 0                                                                                               | Not full Tx FIFO  |
|         |          | 1                                                                                               | Full Tx FIFO      |
| 22      | RFE      | SPI_RFIFO Empty.<br>The SPI_STAT.RFE bit indicates whether the SPI_RFIFO is empty or not empty. |                   |
|         |          | 0                                                                                               | Empty Rx FIFO     |
|         |          | 1                                                                                               | Not empty Rx FIFO |

**Table 25-21:** SPI\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | FCS      | Flow Control Stall Indication.<br>The SPI_STAT.FCS bit indicates whether a slave has asserted the SPI_RDY pin to stall the SPI master while the slave is unable to service the SPI master's request. This bit is valid only when the SPI is a master (SPI_CTL.MSTR =1 and flow control is enabled (SPI_CTL.FCEN =1). |
|         |          | 0 Stall (RDY pin asserted)                                                                                                                                                                                                                                                                                           |
|         |          | 1 No stall (RDY pin de-asserted)                                                                                                                                                                                                                                                                                     |
| 18:16   | TFS      | SPI_TFIFO Status.<br>The SPI_STAT.TFS bits indicate the status of the SPI_TFIFO. The SPI uses this status when evaluating transmit watermark conditions.                                                                                                                                                             |
|         |          | 0 Full TFIFO                                                                                                                                                                                                                                                                                                         |
|         |          | 1 25% empty TFIFO                                                                                                                                                                                                                                                                                                    |
|         |          | 2 50% empty TFIFO                                                                                                                                                                                                                                                                                                    |
|         |          | 3 75% empty TFIFO                                                                                                                                                                                                                                                                                                    |
|         |          | 4 Empty TFIFO                                                                                                                                                                                                                                                                                                        |
| 14:12   | RFS      | SPI_RFIFO Status.<br>The SPI_STAT.RFS bits indicate the status of the SPI_RFIFO. The SPI uses this status when evaluating receive watermark conditions.                                                                                                                                                              |
|         |          | 0 Empty RFIFO                                                                                                                                                                                                                                                                                                        |
|         |          | 1 25% full RFIFO                                                                                                                                                                                                                                                                                                     |
|         |          | 2 50% full RFIFO                                                                                                                                                                                                                                                                                                     |
|         |          | 3 75% full RFIFO                                                                                                                                                                                                                                                                                                     |
|         |          | 4 Full RFIFO                                                                                                                                                                                                                                                                                                         |

**Table 25-21: SPI\_STAT Register Fields (Continued)**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                   |                          |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|
| 11      | TF       | Transmit Finish Indication.<br>The SPI_STAT.TF bit indicates that the SPI has detected the finish of a transmit burst transfer (the SPI_TWC count decrements to zero). This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled.                  |                          |
|         |          | 0                                                                                                                                                                                                                                                                         | No status                |
|         |          | 1                                                                                                                                                                                                                                                                         | Transmit finish detected |
| 10      | RF       | Receive Finish Indication.<br>The SPI_STAT.RF bit indicates that the SPI has detected the finish of a receive burst transfer (the SPI_RWC count decrements to zero). This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.                    |                          |
|         |          | 0                                                                                                                                                                                                                                                                         | No status                |
|         |          | 1                                                                                                                                                                                                                                                                         | Receive finish detected  |
| 9       | TS       | Transmit Start.<br>The SPI_STAT.TS bit indicates that the SPI has detected the start of a transmit burst transfer. A transmit bursts starts with the load of SPI_TWC from the SPI_TWCR. This condition can only occur when SPI_TXCTL.TTI and SPI_TXCTL.TWCEN are enabled. |                          |
|         |          | 0                                                                                                                                                                                                                                                                         | No status                |
|         |          | 1                                                                                                                                                                                                                                                                         | Transmit start detected  |
| 8       | RS       | Receive Start.<br>The SPI_STAT.RS bit indicates that the SPI has detected the start of a receive burst transfer. A receive bursts starts with the load of SPI_RWC from the SPI_RWCR. This condition can only occur when SPI_RXCTL.RTI and SPI_RXCTL.RWCEN are enabled.    |                          |
|         |          | 0                                                                                                                                                                                                                                                                         | No status                |
|         |          | 1                                                                                                                                                                                                                                                                         | Receive start detected   |

**Table 25-21: SPI\_STAT Register Fields (Continued)**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7       | MF       | Mode Fault Indication.<br>The SPI_STAT.MF bit, when SPI is a master and SPI_CTL.PSSE is enabled, indicates that multiple masters have asserted slave select inputs.                                                                                                 |
|         |          | 0                      No status                                                                                                                                                                                                                                    |
|         |          | 1                      Mode fault occurred                                                                                                                                                                                                                          |
| 6       | TC       | Transmit Collision Indication.<br>The SPI_STAT.TC bit, when SPI is a slave, indicates that the load of data into the shift register has occurred too close to the first transmitting edge of the SPI Clock.                                                         |
|         |          | 0                      No status                                                                                                                                                                                                                                    |
|         |          | 1                      Transmit collision occurred                                                                                                                                                                                                                  |
| 5       | TUR      | Transmit Underrun Indication.<br>The SPI_STAT.TUR bit, when the transmit FIFO (SPI_TFIFO) is empty, indicates that the last word in the transmit FIFO has been re-sent as transmit data. Alternately, it indicates that zero has been sent as transmit data.        |
|         |          | 0                      No status                                                                                                                                                                                                                                    |
|         |          | 1                      Transmit underrun occurred                                                                                                                                                                                                                   |
| 4       | ROR      | Receive Overrun Indication.<br>The SPI_STAT.ROR bit, when the receive FIFO (SPI_RFIFO) is full, indicates that a word in the receive FIFO has been overwritten with incoming receive data. Alternately, it indicates that incoming receive data has been discarded. |
|         |          | 0                      No status                                                                                                                                                                                                                                    |
|         |          | 1                      Receive overrun occurred                                                                                                                                                                                                                     |

**Table 25-21:** SPI\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | TUWM     | Transmit Urgent Watermark Breached.<br>The SPI_STAT.TUWM bit indicates that the transmit urgent watermark (SPI_TXCTL.TUWM) has been reached. This condition is cleared when the transmit FIFO fills enough to reach the transmit regular watermark (SPI_TXCTL.TRWM). |
|         |          | 0 TX Regular Watermark reached                                                                                                                                                                                                                                       |
|         |          | 1 TX Urgent Watermark breached                                                                                                                                                                                                                                       |
| 1       | RUWM     | Receive Urgent Watermark Breached.<br>The SPI_STAT.RUWM bit indicates that the receive urgent watermark (SPI_RXCTL.RUWM) has been reached. This condition is cleared when the receive FIFO empties enough to reach the receive regular watermark (SPI_RXCTL.RRWM).   |
|         |          | 0 RX Regular Watermark reached                                                                                                                                                                                                                                       |
|         |          | 1 RX Urgent Watermark breached                                                                                                                                                                                                                                       |
| 0       | SPIF     | SPI Finished.<br>The SPI_STAT.SPIF bit indicates that a single word transfer is complete.                                                                                                                                                                            |
|         |          | 0 No status                                                                                                                                                                                                                                                          |
|         |          | 1 Completed single-word transfer                                                                                                                                                                                                                                     |

## Masked Interrupt Condition Register

The SPI\_ILAT register latches interrupts, queuing the interrupts for service. When a condition is indicated by a bit in the SPI\_STAT register and the corresponding interrupt is unmasked in SPI\_IMSK, the SPI latches the interrupt's bit in SPI\_ILAT.



**SPI\_ILAT: Masked Interrupt Condition Register - R/WE**

Reset = 0x0000 0000

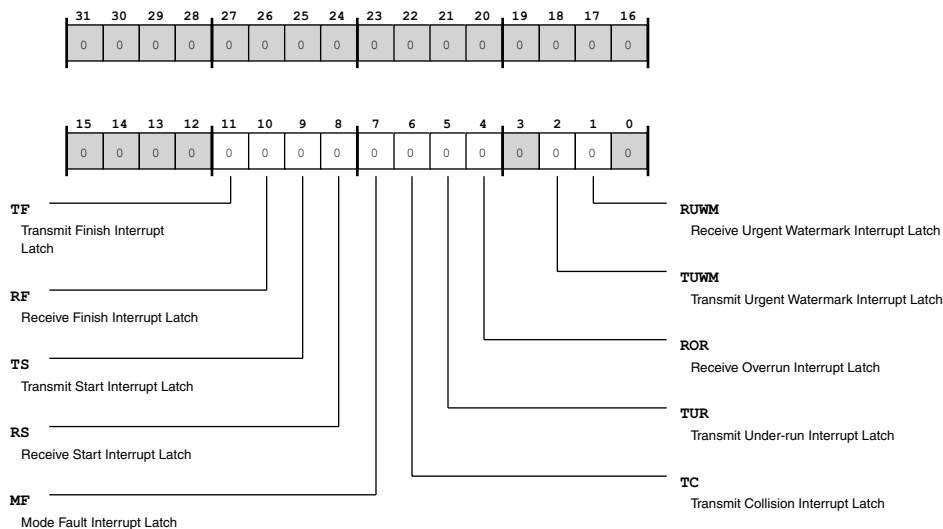


**Figure 25-27: SPI\_ILAT Register Diagram**

**Table 25-22: SPI\_ILAT Register Fields**

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 11      | TF       | Transmit Finish Interrupt Latch. |
|         |          | 0 No interrupt                   |
|         |          | 1 Latched interrupt              |
| 10      | RF       | Receive Finish Interrupt Latch.  |
|         |          | 0 No interrupt                   |
|         |          | 1 Latched interrupt              |
| 9       | TS       | Transmit Start Interrupt Latch.  |
|         |          | 0 No interrupt                   |
|         |          | 1 Latched interrupt              |

**Table 25-22:** SPI\_ILAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                    |                   |
|---------|----------|--------------------------------------------|-------------------|
| 8       | RS       | Receive Start Interrupt Latch.             |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 7       | MF       | Mode Fault Interrupt Latch.                |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 6       | TC       | Transmit Collision Interrupt Latch.        |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 5       | TUR      | Transmit Under-run Interrupt Latch.        |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 4       | ROR      | Receive Overrun Interrupt Latch.           |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 2       | TUWM     | Transmit Urgent Watermark Interrupt Latch. |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |
| 1       | RUWM     | Receive Urgent Watermark Interrupt Latch.  |                   |
|         |          | 0                                          | No interrupt      |
|         |          | 1                                          | Latched interrupt |

## Masked Interrupt Clear Register

The SPI\_ILAT\_CLR register permits clearing individual mask bits in the SPI\_ILAT register without affecting other bits in the register. Use write-1-to-clear on a bit in SPI\_ILAT\_CLR to clear the corresponding bit in the SPI\_ILAT register.

### SPI\_ILAT\_CLR: Masked Interrupt Clear Register - R/W

Reset = 0x0000 0000

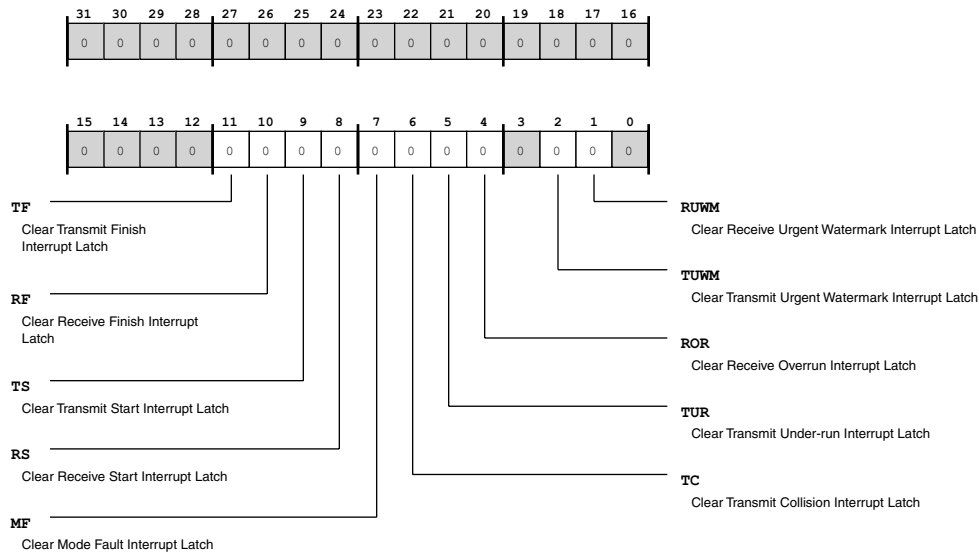


**Figure 25-28:** SPI\_ILAT\_CLR Register Diagram

**Table 25-23:** SPI\_ILAT\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                   |
|---------|----------|-------------------------------------------|
| 11      | TF       | Clear Transmit Finish Interrupt Latch.    |
| 10      | RF       | Clear Receive Finish Interrupt Latch.     |
| 9       | TS       | Clear Transmit Start Interrupt Latch.     |
| 8       | RS       | Clear Receive Start Interrupt Latch.      |
| 7       | MF       | Clear Mode Fault Interrupt Latch.         |
| 6       | TC       | Clear Transmit Collision Interrupt Latch. |
| 5       | TUR      | Clear Transmit Under-run Interrupt Latch. |
| 4       | ROR      | Clear Receive Overrun Interrupt Latch.    |

Table 25-23: SPI\_ILAT\_CLR Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                          |
|---------|----------|--------------------------------------------------|
| 2       | TUWM     | Clear Transmit Urgent Watermark Interrupt Latch. |
| 1       | RUWM     | Clear Receive Urgent Watermark Interrupt Latch.  |

Receive FIFO Data Register

The SPI\_RFIFO register has an interface to the receive shift register in the SPI and has an interface to the processor's data buses. The top level of the buffer is visible to programs as the 32-bit SPI\_RFIFO register, but the size (number of word locations) of the receive FIFO is actually flexible with transfer word size. The size of the receive FIFO is 8 if word size is 8-bit, or the size is 4 if word size is 16-bit, or the size is 2 if word size is 32-bit.

Both masters and slaves may stop or stall receive transfers based on FIFO status. When the receive FIFO is full, the SPI master stops initiating new transfers on the SPI if SPI\_RXCTL.RTI is enabled. A slave may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data reception continues after SPI\_RFIFO is full, the data in the receive FIFO is invalid, and the SPI indicates this condition with receive overrun (SPI\_STAT.ROR). This condition is possible when SPI\_RXCTL.RTI =0 and SPI\_RXCTL.REN =1 for a master, or for a slave that does not exercise flow control.

Note that the receive FIFO is reset (cleared) when the SPI is disabled after being enabled.

SPI\_RFIFO: Receive FIFO Data Register - R/WE

Reset = 0x0000 0000

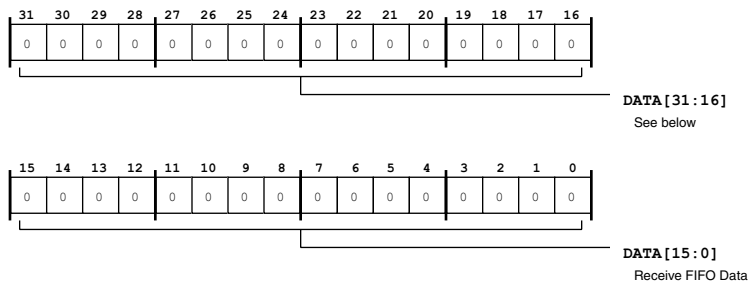


Figure 25-29: SPI\_RFIFO Register Diagram

Table 25-24: SPI\_RFIFO Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | DATA     | Receive FIFO Data.      |

## Transmit FIFO Data Register

The `SPI_TFIFO` register has an interface to the transmit shift register in the SPI and has an interface to the processor's data buses. The top level of the buffer is visible to programs as the 32-bit `SPI_TFIFO` register, but the size (number of word locations) of the transmit FIFO is actually flexible with transfer word size. The size of the transmit FIFO is 8 if word size is 8-bit, or the size is 4 if word size is 16-bit, or the size is 2 if word size is 32-bit.

Both masters and slaves may stop or stall transmit transfers based on FIFO status. When the transmit FIFO is empty, the SPI master stops initiating new transfers on the SPI if `SPI_TXCTL.TTI` is enabled. A slave may stall the SPI interface when the content of the FIFO crosses the selected watermark. If data transmit requests continue after `SPI_TFIFO` is empty, the data sent from the transmit FIFO is invalid, and the SPI indicates this condition with transmit underrun (`SPI_STAT.TUR`). This condition is possible when `SPI_TXCTL.TTI = 0` and `SPI_TXCTL.TEN = 1` for a master, or for a slave that does not exercise flow control.

Note that the transmit FIFO is reset (cleared) when the SPI is disabled after being enabled.

**SPI\_TFIFO: Transmit FIFO Data Register - R/W**

Reset = 0x0000 0000

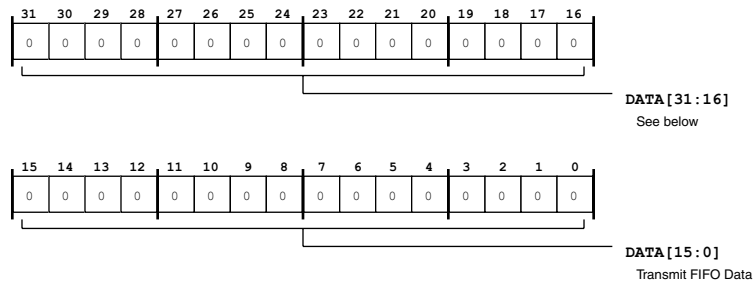


**Figure 25-30:** SPI\_TFIFO Register Diagram

**Table 25-25:** SPI\_TFIFO Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | DATA     | Transmit FIFO Data.     |



## 26 Serial Port (SPORT)

Unlike the SPI interface, which has been designed for SPI-compatible communication only, the serial ports (SPORTs) support a variety of serial data communication protocols. In addition, the SPORTs provide a glueless hardware interface to many industry-standard data converters and codecs. With support for high data rates and dual half-duplex data paths, the SPORT interface is a perfect choice for direct serial interconnection between two or more processors in a multiprocessor system. Many processors provide compatible serial interfaces, including DSPs from Analog Devices and other manufacturers.

The SPORT top module comprises of two half SPORTs with identical functionality. Each SPORT half can be independently configured as either a transmitter or receiver and can be coupled with the other HSPORT within the same SPORT. Further, each SPORT half provides two synchronous half-duplex data lines to double the total supported data streams.

Each SPORT half has the same capabilities and is programmed in the same way. The interface specifications of each SPORT half are shown in the following table.

**Table 26-1:** SPORT Specifications

| Feature                                   | Availability                                                                                |
|-------------------------------------------|---------------------------------------------------------------------------------------------|
| Connectivity                              |                                                                                             |
| Multiplexed Pinout                        | Yes                                                                                         |
| Internal connections between SPORT halves | Yes. Only Clock and/or Frame Sync can be loopbacked internally between paired SPORT halves. |
| Interrupt Control                         | Yes                                                                                         |
| Protocol                                  |                                                                                             |
| Master Capable                            | Yes                                                                                         |
| Slave Capable                             | Yes                                                                                         |
| Transmission Simplex                      | Yes                                                                                         |
| Transmission Half Duplex                  | Yes                                                                                         |
| Transmission Full Duplex                  | No. The paired SPORT halves can however, be effectively used for full-duplex communication. |

**Table 26-1:** SPORT Specifications (Continued)

| Feature          | Availability                                                                |
|------------------|-----------------------------------------------------------------------------|
| Access Type      |                                                                             |
| Data Buffer      | Yes. Each SPORT half has its own set of control registers and data buffers. |
| Core Data Access | Yes                                                                         |
| DMA Data Access  | Yes                                                                         |
| DMA Channels     | One per SPORT half                                                          |
| DMA Chaining     | Yes                                                                         |
| Boot Capable     | No                                                                          |
| Local Memory     | No                                                                          |
| Clock Operation  | See data sheet                                                              |

## Features

An individual SPORT module consists of two independently configurable SPORT halves with identical functionality. These SPORT halves offer the following features.

- Two bidirectional data lines—Primary (0) and Secondary (1) per SPORT half, configurable as either transmitters or receivers. Therefore, each SPORT half can be configured for two transmitter or two receiver channels, permitting two unidirectional streams into or out of the same SPORT half. This bidirectional functionality provides greater flexibility for serial communications. Further, two SPORT halves can be combined to enable full-duplex, dual-stream communications.
- Six operation modes
  - a. Standard DSP serial mode
  - b. I<sup>2</sup>S mode
  - c. Left-Justified mode
  - d. Right-Justified Mode
  - e. Multichannel Mode
  - f. Packed mode



- Improved granularity for internal clock generation, allowing both even and odd SCLK to SPORT\_CLK ratios. If both data lines of a SPORT half are active, it can have a maximum throughput of 2 x SPORT\_CLK. The SPORTs can accept an input clock from an external source.
- Configurable rising or falling edge of the SPORT\_CLK for driving or sampling data and frame sync.
- Gated clock mode support for both internal clock and external clock mode in DSP serial mode and stereo modes (Left-justified and I<sup>2</sup>S mode).
- Operates with or without a frame synchronization signal for each data word, with internally generated or externally generated frame signals, with active high or active low frame signals, and with either of two configurable pulse widths and frame signal timing.
- Status flagging and optional interrupt generation for prematurely received external frame syncs.
- External frame sync signal can be configured as level-sensitive or edge-sensitive signal.
- Serial data words between 4 and 32 bits in length, either in most significant bit (MSB) first or in least significant bit (LSB) first format. Optional sign-extension on received data.
- Optional 16-bit to 32-bit word packing when SPORT is configured as receiver and 32-bit to 16-bit word unpacking when configured as Transmitter.
- When configured as transmitter, both primary and secondary data paths can have optional compress engines enabled. Similarly, in receiver mode, both paths can have optional have expand engines enabled. A-law and  $\mu$ -law compression/decompression hardware companding according to G.711 specification on transmitted and received words in all operating modes.
- Status flagging and optional interrupt generation for Transmit under-run or Receive over-flow.
- Supports multichannel mode for TDM interfaces. Each SPORT half can transmit or receive data selectively from a time-division-multiplexed serial bit stream on 128 contiguous channels from a stream of up to 1024 total channels. This mode can be useful for H.100/H.110 and other telephony interfaces as a network communication scheme for multiple processors.
- Performs interrupt-driven, single word transfers to and from on-chip or off-chip memory under processor control.
- Dedicated DMA channel for each SPORT half. This DMA is common for both data lines and can be configured for multiple work units such as auto-buffer based (for a repeated, identical range of transfers) or descriptor-based (individual or repeated ranges of transfers with differing DMA parameters).
- SPORT DMA's can be programmed to accept the incoming trigger when configured as Trigger Slave and are capable of generating outgoing trigger as well.
- When using DMA in transmit mode, a Transfer Finish Interrupt (TFI) can be used to make sure that the last word of the transfer has been shifted out of the transmit shift register.
- SPMUX, a local multiplexing block integrated between the SPORT and the PinMux logic, provides the ability to route and share the clocks and/or frame sync between the SPORT halves of the SPORT

module. The internal routing helps to reduce the total number of processor pins required for the interface. This is especially efficient when a SPORT is used for full-duplex data transfers.

- Interface with the ADC Control Module (ACM) block. This allows the frame sync and clock signals generated from ACM block to be routed internally to one of the SPORT halves of SPORT1.

## Signal Descriptions

Each SPORT half module has five dedicated pins, as described in the following table. The actual pin name varies with different SPORT halves. The individual SPORT half does not share any of its pins across the pair. However, if required, clock and frame sync signals can be interconnected between the SPORT half pair, as explained in SPORT pin MUX section.

All the SPORT signals are available on Port B, D, E, G and are multiplexed with other peripheral signals. By default, these pins are in GPIO mode. To enable the pins for SPORT functionality, the appropriate bits must be set in the `PORTx_FER` and `PORTx_MUX` registers. It is advised to configure `PORTx_MUX` register before `PORTx_FER`.

**Table 26-2:** SPORT Pin Descriptions

| Internal Node | Direction | Description                                                                                                                                                                                                                                                            |
|---------------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SPORT_CLK     | I/O       | Transmit/Receive Serial Clock. Data and Frame Sync are driven/sampled with respect to this clock. This signal can be either internally or externally generated.                                                                                                        |
| SPORT_FS      | I/O       | Transmit/Receive Frame Sync. The frame sync pulse initiates shifting of serial data. This signal is either generated internally or externally.                                                                                                                         |
| SPORT_D0      | I/O       | Transmit/receive Primary Data channel. Bidirectional data pin. This signal can be configured as an output to transmit serial data, or as an input to receive serial data.                                                                                              |
| SPORT_D1      | I/O       | Transmit/receive secondary Data channel. Bidirectional data pin. This signal can be configured as an output to transmit serial data, or as an input to receive serial data.                                                                                            |
| SPORT_TDV     | O         | Multichannel Transmit Data Valid. This signal is only active if SPORT is configured in multichannel transmit mode. The signal is asserted during enabled slots based on the channel selection registers ( <code>SPORT_CS0_A</code> through <code>SPORT_CS3_B</code> ). |

The signals are known as Transmit signal when the serial port is configured in transmit mode (`SPORT_CTL_A.SPTRAN = 1`); while are known as Receive signals when configured in receive mode (`SPORT_CTL_A.SPTRAN = 0`). These SPORT signals are described in the sections below.

## Serial Clock

The serial port clock (SPORT\_ACLK) signal is considered a Receive serial clock if the transfer direction is configured as receiver; while it is considered a Transmit serial clock when configured as transmitter.

The serial clock (SPORT\_ACLK) is one of the control signal of serial port depending on which the data bits are shifted-in or shifted-out serially based on the direction selected. The frame sync signal is also driven (in internal frame sync mode) or sampled (in external frame sync mode) with respect to serial clock signal. The serial clock can be internally generated from processor's system clock (SCLK1) or externally provided, based on SPORT\_CTL\_A.ICLK bit setting.

If a SPORT is configured in internal clock mode (SPORT\_CTL\_A.ICLK = 1), then the SPORT\_DIV\_A.CLKDIV field specifies the divider to generate serial port clock signal from its fundamental clock, SCLK1. This divisor is a 16-bit value, allowing a wide range of serial clock rates. Use the following equation to calculate the serial clock frequency:

$$\text{SPORT\_ACLK} = [\text{SCLK1} \div (\text{SPORT\_DIV\_A.CLKDIV} + 1)]$$

Use the following equation to determine the value of SPORT\_DIV\_A.CLKDIV, given the SCLK1 frequency and desired serial port clock frequency:

$$\text{SPORT\_DIV\_A.CLKDIV} = [(\text{SCLK1} \div \text{SPORT\_ACLK}) - 1]$$

This equation results in improved granularity for internal clock generation, allowing both odd and even SCLK1:SPORT\_ACLK ratios.

It also supports 1:1 SPORT\_ACLK to SCLK1 ratio, when CLKDIV field is programmed to zero, resulting in serial port clock frequency equal to system clock. But caution must be exercised not to exceed the maximum SPORT\_ACLK frequency specified in the data sheet. Therefore if SCLK1 is greater than the data sheet limit, SPORT\_ACLK:SCLK1 ratio must be limited to 1:2. For other SCLK1 frequencies, this ratio can be programmed up to 1:1.

In certain operating modes, the serial port can be configured to generate gated clock which is active only for the duration of valid data. In some applications, it can be used to generate a general-purpose clock in the system. In this case the SPORT must be enabled with appropriate SPORT\_DIV\_A.CLKDIV divisor field in internal clock mode.

If a SPORT is configured in external clock mode (SPORT\_CTL\_A.ICLK = 0), then serial clock is a input signal making the SPORT to operate in slave mode. The SPORT\_DIV\_A.CLKDIV is ignored. The optional loopback capability provided by SPMUX block, allows slave SPORT to use the serial clock from the neighboring serial port.

The external serial clock can be of the order of system clock1, considering 83 MHz as the maximum SPORT clock frequency. Note that externally supplied serial clock need not be in synchronous with processor system clock. Further, the external clock can be a gated clock but it must comply the requirements described in Gated Clock Mode section. Please refer appropriate product data sheet for exact a.c. timing specifications.

## Frame Sync

The serial port frame sync (SPORT\_AFS) signal is considered a Receive Frame Sync if the transfer direction is configured as receiver; while it is considered a Transmit Frame Sync when configured as transmitter.

Frame sync is also a control signal, generally used to determine the start of new word or frame. Upon detecting this signal, serial port starts shifting in or out the new data bits serially based on the direction selected. The frame sync signal can be internally generated from its serial clock (SPORT\_ACLK) or externally provided, based on the SPORT\_CTL\_A.IFS bit setting.

If SPORT is configured for internal frame sync mode (SPORT\_CTL\_A.IFS = 1), then the SPORT\_DIV\_A.FSDIV field specifies the divider to generate SPORT\_AFS signal from the serial clock. This divisor is a 16-bit value, allowing a wide range of frame sync rates to initiate periodic transfers. The serial clock may be internally generated or externally supplied and it is counted equal to divisor specified before a frame sync pulse is generated. The formula for the number of cycles between frame sync pulses is:

Number of serial clocks between frame syncs = (SPORT\_DIV\_A.FSDIV + 1)

Use the following equation to determine the value of SPORT\_DIV\_A.FSDIV, given the serial clock frequency and desired frame sync frequency:

$$\text{SPORT\_DIV\_A.FSDIV} = [(\text{SPORT\_ACLK} \div \text{SPORT\_AFS}) - 1]$$

The frame sync is continuously active when SPORT\_DIV\_A.FSDIV = 0. The value of SPORT\_DIV\_A.FSDIV should not be less than the serial word length minus one (the value of the SPORT\_CTL\_A.SLEN bit field), as this may cause an external device to abort the current operation or cause other unpredictable results. *After enabling the SPORT, the first internal frame sync appears after a delay of (SPORT\_DIV\_A.FSDIV + 3) serial clocks.*

If a SPORT is configured in external frame sync mode (SPORT\_CTL\_A.IFS = 0), then SPORT\_AFS is a input signal and the SPORT\_DIV\_A.FSDIV field of the SPORT\_DIV\_A register is ignored. By default, this external signal is level-sensitive, but can be configured as an edge-sensitive signal by setting SPORT\_CTL\_A.FSED bit. The frame sync is expected to be synchronous with the serial clock. If not, it must meet the timing requirements that appear in the product specific data sheet.

The serial port can be used as a counter for dividing an external clock to generate periodic pulses or periodic interrupts. The SPORT must be enabled with appropriate SPORT\_DIV\_A.FSDIV divisor field in external clock, internal data-independent frame sync mode.

In some of the operating modes, the serial port can be programmed to treat the frame sync signal as an optional signal by clearing the SPORT\_CTL\_A.FSR bit (it can be used to start the continuous transfers and subsequently ignored). Characteristics of the frame sync depend on the settings in the SPORT control registers and the SPORT's operating mode. For more information, refer to the SPORT control register bits and respective operating mode details.

## Data Signals

Each SPORT half has two bi-directional data lines known as the primary transmit or receive data channel (SPORT\_ADO) and the secondary transmit or receive data channel (SPORT\_AD1). Both the data lines can be configured as either transmitters or receivers using the SPORT\_CTL\_A.SPTRAN bit, permitting dual unidirectional data streams to increase the data throughput of the serial port.

Both data lines can be individually enabled or disabled using the SPORT\_CTL\_A.SPENPRI and the SPORT\_CTL\_A.SPENSEC bits. However, if using both, it is advised to enable or disable them concurrently. They do not behave as totally separate SPORTs; rather, they operate in a synchronous manner (sharing a clock and frame sync) but on separate data paths. All of the SPORT control settings are common for both channels but the single DMA channel per serial half serves both primary and secondary data channels. Also, both primary and secondary channels have separate data buffers, shift registers and optional companding logic in their path.

When a serial port is configured in multichannel transmit mode, the data pins three-states during inactive channel slots. This allows multiple serial port transmitters to operate on the same bus with different active channels.

See the Architecture section for more details about data transfer operation.

## Transmit Data Valid Signal

The Transmit Data Valid (SPORT\_ATDV) signal is available only in multichannel modes (including packed mode) of a SPORT configured as a transmitter. This signal is active during transmission of enabled multichannel slots and remains in an inactive state for the disabled channels. In other words, the SPORT\_ATDV signal is active whenever a serial port is driving the data pins and stays inactive when the data pins three-states. Therefore the SPORT\_ATDV signal can serve as an output-enable signal for the data transmit pin.

## Functional Description

The following section provides general information about functionality of the serial ports of processors.

- [\*Architectural Concepts\*](#)
- [\*Data Types and Companding\*](#)
- [\*Transmit Path\*](#)
- [\*Receive Path\*](#)

## SPORT Register List

The serial port SPORT controller, with its range of clock and frame synchronization options, supports a variety of serial communication protocols and provides a glue-less hardware interface to many industry-standard data converters and CODECs. Each SPORT has two independent halves (A and B), and each half contains two channels (primary and secondary). A set of registers govern SPORT operations. For more information on SPORT functionality, see the SPORT register descriptions.

**Table 26-3:** SPORT Module Registers

| Name          | Description                                         |
|---------------|-----------------------------------------------------|
| SPORT_CTL_A   | Half SPORT 'A' Control Register                     |
| SPORT_DIV_A   | Half SPORT 'A' Divisor Register                     |
| SPORT_MCTL_A  | Half SPORT 'A' Multi-channel Control Register       |
| SPORT_CS0_A   | Half SPORT 'A' Multi-channel 0-31 Select Register   |
| SPORT_CS1_A   | Half SPORT 'A' Multi-channel 32-63 Select Register  |
| SPORT_CS2_A   | Half SPORT 'A' Multi-channel 64-95 Select Register  |
| SPORT_CS3_A   | Half SPORT 'A' Multi-channel 96-127 Select Register |
| SPORT_ERR_A   | Half SPORT 'A' Error Register                       |
| SPORT_MSTAT_A | Half SPORT 'A' Multi-channel Status Register        |
| SPORT_CTL2_A  | Half SPORT 'A' Control 2 Register                   |
| SPORT_TXPRI_A | Half SPORT 'A' Tx Buffer (Primary) Register         |
| SPORT_RXPRI_A | Half SPORT 'A' Rx Buffer (Primary) Register         |
| SPORT_TXSEC_A | Half SPORT 'A' Tx Buffer (Secondary) Register       |
| SPORT_RXSEC_A | Half SPORT 'A' Rx Buffer (Secondary) Register       |
| SPORT_CTL_B   | Half SPORT 'B' Control Register                     |
| SPORT_DIV_B   | Half SPORT 'B' Divisor Register                     |
| SPORT_MCTL_B  | Half SPORT 'B' Multi-channel Control Register       |
| SPORT_CS0_B   | Half SPORT 'B' Multi-channel 0-31 Select Register   |

**Table 26-3:** SPORT Module Registers (Continued)

| Name          | Description                                        |
|---------------|----------------------------------------------------|
| SPORT_CS1_B   | Half SPORT 'B' Multi-channel 32-63 Select Register |
| SPORT_CS2_B   | Half SPORT 'B' Multichannel 64-95 Select Register  |
| SPORT_CS3_B   | Half SPORT 'B' Multichannel 96-127 Select Register |
| SPORT_ERR_B   | Half SPORT 'B' Error Register                      |
| SPORT_MSTAT_B | Half SPORT 'B' Multi-channel Status Register       |
| SPORT_CTL2_B  | Half SPORT 'B' Control 2 Register                  |
| SPORT_TXPRI_B | Half SPORT 'B' Tx Buffer (Primary) Register        |
| SPORT_RXPRI_B | Half SPORT 'B' Rx Buffer (Primary) Register        |
| SPORT_TXSEC_B | Half SPORT 'B' Tx Buffer (Secondary) Register      |
| SPORT_RXSEC_B | Half SPORT 'B' Rx Buffer (Secondary) Register      |

## ADSP-BF60x SPORT Interrupt List

**Table 26-4:** ADSP-BF60x SPORT Interrupt List

| Description             | ID | DMA Channel | Sensitivity |
|-------------------------|----|-------------|-------------|
| SPORT0 Channel A DMA    | 43 | 0           | LEVEL       |
| SPORT0 Channel A Status | 44 |             | LEVEL       |
| SPORT0 Channel B DMA    | 45 | 1           | LEVEL       |
| SPORT0 Channel B Status | 46 |             | LEVEL       |
| SPORT1 Channel A DMA    | 47 | 2           | LEVEL       |
| SPORT1 Channel A Status | 48 |             | LEVEL       |
| SPORT1 Channel B DMA    | 49 | 3           | LEVEL       |
| SPORT1 Channel B Status | 50 |             | LEVEL       |

**Table 26-4:** ADSP-BF60x SPORT Interrupt List (Continued)

| Description             | ID | DMA Channel | Sensitivity |
|-------------------------|----|-------------|-------------|
| SPORT2 Channel A DMA    | 51 | 4           | LEVEL       |
| SPORT2 Channel A Status | 52 |             | LEVEL       |
| SPORT2 Channel B DMA    | 53 | 5           | LEVEL       |
| SPORT2 Channel B Status | 54 |             | LEVEL       |

## ADSP-BF60x SPORT Trigger List

**Table 26-5:** Masters ADSP-BF60x SPORT Trigger List

| Description          | ID | Sensitivity |
|----------------------|----|-------------|
| SPORT0 Channel A DMA | 20 | PULSE/EDGE  |
| SPORT0 Channel B DMA | 21 | PULSE/EDGE  |
| SPORT1 Channel A DMA | 22 | PULSE/EDGE  |
| SPORT1 Channel B DMA | 23 | PULSE/EDGE  |
| SPORT2 Channel A DMA | 24 | PULSE/EDGE  |
| SPORT2 Channel B DMA | 25 | PULSE/EDGE  |
|                      |    |             |

**Table 26-6:** Slaves ADSP-BF60x SPORT Trigger List

| Description          | ID | Sensitivity |
|----------------------|----|-------------|
| SPORT0 Channel A DMA | 20 |             |
| SPORT0 Channel B DMA | 21 |             |
| SPORT1 Channel A DMA | 22 |             |
| SPORT1 Channel B DMA | 23 |             |



**Table 26-6:** Slaves ADSP-BF60x SPORT Trigger List (Continued)

| Description          | ID | Sensitivity |
|----------------------|----|-------------|
| SPORT2 Channel A DMA | 24 |             |
| SPORT2 Channel B DMA | 25 |             |
|                      |    |             |

## Block Diagram

The serial port is configured in transmit mode, if `SPORT_CTL_A.SPTRAN` control bit is set. If this bit is cleared, serial port configures in receive mode. If `SPORT_CTL_A.SPENPRI` control bit is set, then serial port activates primary transmit/receive path. If `SPORT_CTL_A.SPENSEC` control bit is set, then it activates secondary transmit/receive path. Both data channels can be enabled to allow synchronous dual-stream communication. Each path optionally supports Hardware companding or expanding as well. Once a path is activated, data is shifted in response to a frame sync at the rate of serial clock. Inactive data buffers are not used and should not be accessed. An application program must use the appropriate data buffers.

These serial ports are not UARTs and cannot communicate with an RS-232 device or any other asynchronous communications protocol. One way to implement RS-232 compatible communication with the processor is to use two of the FLAG pins as asynchronous data receive and transmit signals.

The following figure shows a detailed block diagram of a SPORT half side.

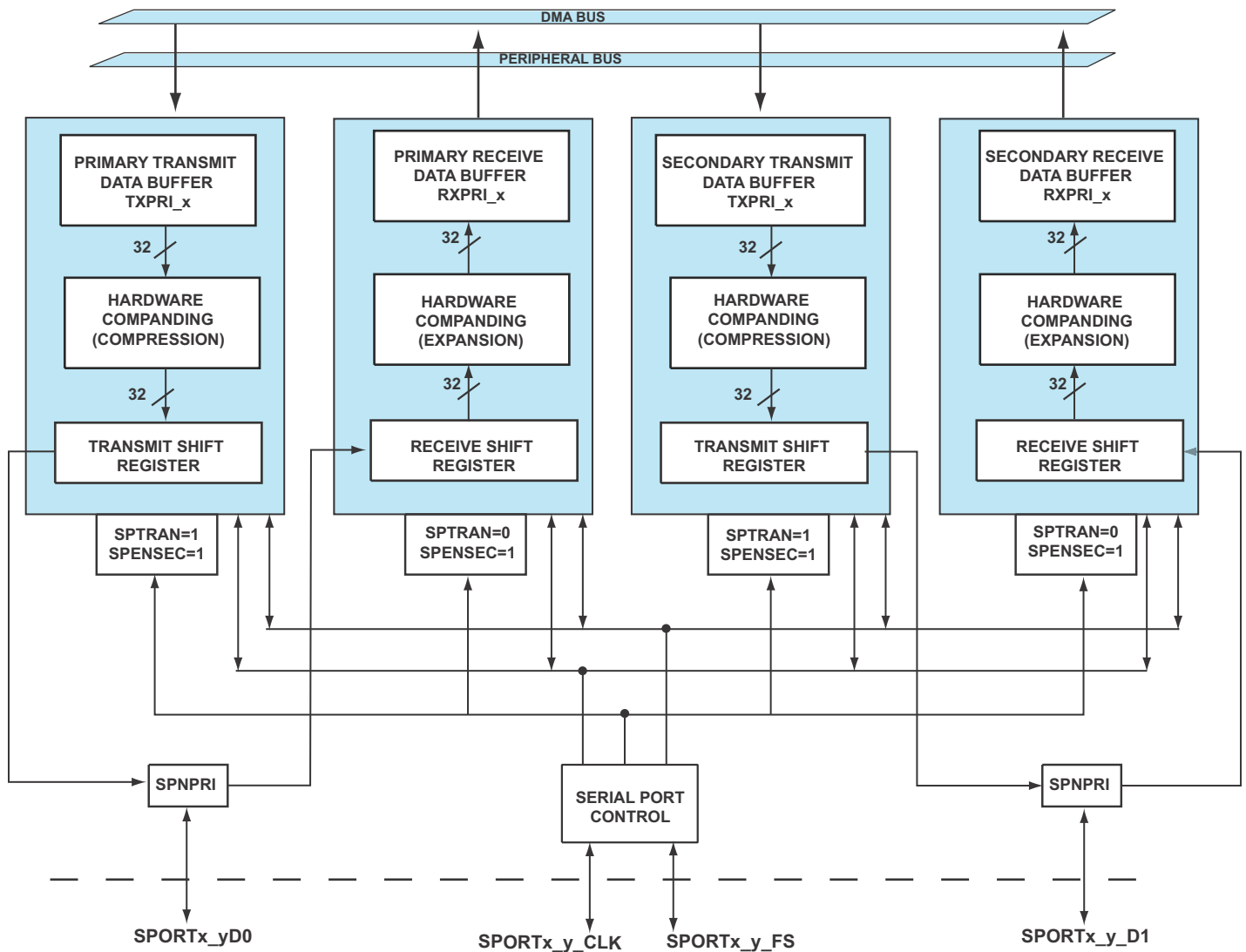


**Figure 26-1:** Serial Port Block Diagram

## Architectural Concepts

Each SPORT module consists of two separate blocks, known as half-SERIAL PORT (HSPORT) A and B, with identical functionality. These blocks can be independently configurable as either transmitter or receiver; and optionally coupled together internally in a limited way. Each HSPORT also supports two synchronous bidirectional data paths, referred as primary (D0) and secondary (D1) data lines, as shown in the following figure.

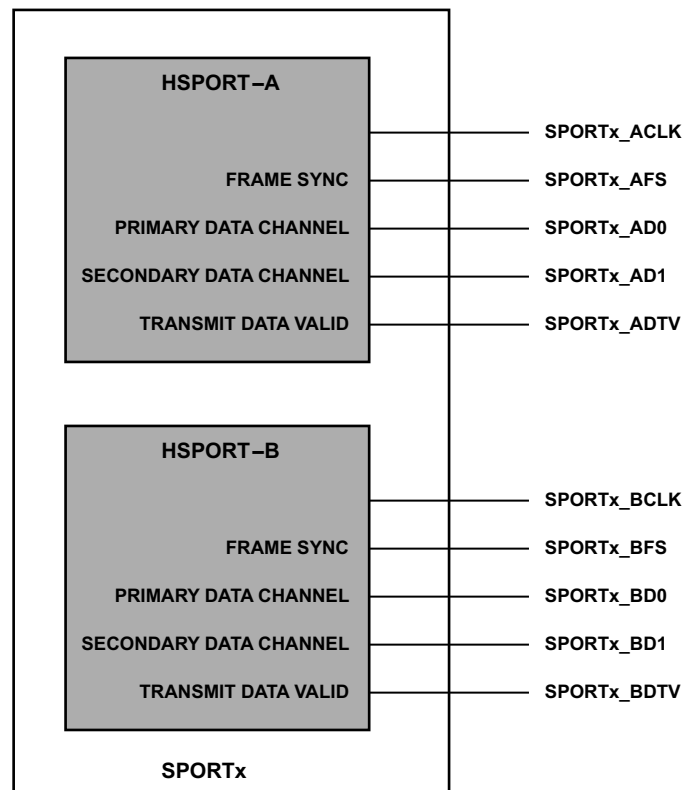


**Figure 26-2:** Top Level SPORT Diagram

Each HSPORT can be configured as either transmitter or receiver, according to which the pair of data signals transmit or receive data bits synchronously. The `SPORT_CTL_A.SPTRAN` bit controls the direction for both data paths of the HSPORT. Each HSPORT has its own set of control registers and data buffers grouped per SPORT module. The dual data signals of each HSPORT cannot transmit and receive the data simultaneously for full-duplex operation. Two HSPORTs must be combined to achieve full-duplex operation.

Serial communications are synchronized to the serial clock signal. Every data bit must be accompanied by a clock pulse. Each serial port can internally generate its own serial clock signal from the processor's system clock using the divisor field of the `SPORT_DIV_A.FSDIV` bit field. If programmed, serial ports can also operate in external clock mode. Both primary and secondary data channels shift data based on `SPORT_CLK` rate and the `SPORT_CTL_A.CKRE` bit

In addition to the serial clock signal, data may be signaled by a frame synchronization signal. The framing signal can occur at the beginning of an individual word or at the beginning of a block of words. The configuration of frame sync signal depends upon the type of serial device connected to the processor. Each serial port can generate its own frame sync signal (`SPORT_FS`) depending on the bit settings of SPORT control register. An internally generated frame sync is derived from the SPORT clock using the divisor field of the `SPORT_DIV_A.FSDIV` bit field. Serial ports can also accept external `SPORT_FS` signal. Both primary and secondary data paths starts shifting data after detecting a valid frame sync signal according to control bit settings and operating mode of serial port. A variety of serial data communication protocols can be

emulated according to the frame sync format. All frame sync options are available whether the signal is generated internally or externally generated.

## Multiplexer Logic

There is a local muxing block, known as SPMUX, that is integrated between the SPORT and the PinMux logic of processor. It allows flexibility to route and share the clock and frame sync signals between the SPORT half pair of a SPORT. This is where the two independent SPORT halves of a SPORT can be coupled together. This feature can be used to reduce the total number of pins for the interface and is considered to be efficient when the SPORT half pair is used for full-duplex operation.

The SPORT\_CTL2\_A register is used to configure this loopback feature. The control bits of this register are as described in the "Register Descriptions" section of this chapter.

The multiplexing depends on the SPORT\_CTL\_A.IFS and SPORT\_CTL\_A.ICLK bit settings of the and is further controlled by the SPORT\_CTL2\_A.CKMUXSEL and SPORT\_CTL2\_A.FSMUXSEL bit settings of the SPORT half pair. The following two tables show the valid combinations for the SPORT\_CTL\_A.IFS, SPORT\_CTL\_A.ICLK, SPORT\_CTL2\_A.CKMUXSEL and SPORT\_CTL2\_A.FSMUXSEL bit settings. All other settings are considered to be illegal. The illegal settings, however, are not checked or prevented by hardware. Programs should ensure that only legal combinations are used.

The table's Routing column uses the following abbreviations.

- HSx\_FI = Frame sync input. It can be provided by external device or by the neighboring SPORT half.
- HSx\_FO = Frame sync output. When SPORT is configured in internal frame sync mode.
- SPx\_FS = signal appearing on frame sync pin of the SPORT half.

*In the tables, the Half-SPORT pair of a SPORT, A and B, are referred as HS0 and HS1 and are applicable for all SPORTs.*

**Table 26-7:** Frame Sync Combinations

| FS Combination ID | HS0_IFS | HS1_IFS | FS0MUX | FS1MUX | Routing                             |
|-------------------|---------|---------|--------|--------|-------------------------------------|
| 1                 | 0       | 0       | 0      | 0      | Native FS Operation                 |
| 2                 | 0       | 1       | 0      | 0      | Native FS Operation                 |
| 3                 | 1       | 0       | 0      | 0      | Native FS Operation                 |
| 4                 | 1       | 1       | 0      | 0      | Native FS Operation                 |
| 5                 | 0       | 0       | 1      | 0      | HS0_FI ≤ SP1_FS;<br>HS1_FI ≤ SP1_FS |

**Table 26-7:** Frame Sync Combinations (Continued)

| FS Combination ID | HS0_IFS | HS1_IFS | FS0MUX | FS1MUX | Routing                                            |
|-------------------|---------|---------|--------|--------|----------------------------------------------------|
| 6                 | 0       | 1       | 1      | 0      | $HS0\_FI \leq HS1\_FO \geq SP1\_FS$                |
| 7                 | 0       | 0       | 0      | 1      | $HS1\_FI \leq SP0\_FS$ ;<br>$HS0\_FI \leq SP0\_FS$ |
| 8                 | 1       | 0       | 0      | 1      | $HS1\_FI \leq HS0\_FO \geq SP0\_FS$                |

The table's Routing column uses the following abbreviations.

- $HSx\_CI$  = serial clock input. It can be provided by external device or by neighboring SPORT half.
- $HSx\_CO$  = serial clock output. When SPORT is configured in internal clock mode.
- $SPx\_CLK$  = signal appearing on serial clock pin of the SPORT half.

**Table 26-8:** Clock Combinations

| CLK Combination ID | HS0_ICLK | HS1_ICLK | CK0MUX | CK1MUX | Routing                                              |
|--------------------|----------|----------|--------|--------|------------------------------------------------------|
| 9                  | 0        | 0        | 0      | 0      | Native CLK Operation                                 |
| 10                 | 0        | 1        | 0      | 0      | Native CLK Operation                                 |
| 11                 | 1        | 0        | 0      | 0      | Native CLK Operation                                 |
| 12                 | 1        | 1        | 0      | 0      | Native CLK Operation                                 |
| 13                 | 0        | 0        | 1      | 0      | $HS0\_CI \leq SP1\_CLK$ ;<br>$HS1\_CI \leq SP1\_CLK$ |
| 14                 | 0        | 1        | 1      | 0      | $HS1\_CI \leq HS1\_CO \geq SP1\_CLK$                 |
| 15                 | 0        | 0        | 0      | 1      | $HS1\_CI \leq SP0\_CLK$ ;<br>$HS0\_CI \leq SP0\_CLK$ |
| 16                 | 1        | 0        | 0      | 1      | $HS1\_CI \leq HS0\_FO \geq SP0\_CLK$                 |

The following additional points on these combination settings should be noted.

- FS IDs 1–4 are supported with all CLK IDs 9–16.
- FS ID 5 is only supported with CLK ID 13 and vice-versa.

- FS ID 6 is only supported with CLK ID 14 and vice-versa.
- FS ID 7 is only supported with CLK ID 15 and vice-versa.
- FS ID 8 is only supported with CLK ID 16 and vice-versa.
- CLK IDs 9–12 are supported with all FS IDs 1–8.

**NOTE:** From these tables, one can note that a SPORT half can import serial clock signal from paired HSPORT, only when it configured in external clock mode; similarly, it can import frame sync signal, only when it is configured in external frame sync mode. The neighboring SPORT may be master (generates it's own serial clock or frame sync signal) or slave (accepting external clock or external frame sync). It can be also noticed that, SPORT\_CTL2 register programming is required only at the acceptor SPORT half side; and not required at the donor SPORT half side to enable this sharing.

Note that Polarity bits such as SPORT\_CTL\_A.CKRE and SPORT\_CTL\_A.LFS should have identical settings when using muxing between two SPORT halves.

**NOTE:** There are some limitations on use of SPMUX logic when a SPORT is interfaced with an ACM module of the processor. Refer to the ACM chapter for more details.

## Data Types and Companding

The Data Type select field SPORT\_CTL\_A.DTYPE bit specifies one of the four data formats supported by serial ports. These formats can be used in any of the operating mode of serial port.

**Table 26-9:** Data Type Bit Field Settings

| DTYPE field | SPORT Receiver                          | SPORT Transmitter    |
|-------------|-----------------------------------------|----------------------|
| 00          | Right-Justify, zero-fill unused MSB's   | Normal Operation     |
| 01          | Right-Justify, sign-extend unused MSB's | Reserved             |
| 10          | Expand using u-law                      | Compress using u-law |
| 11          | Expand using A-law                      | Compress using A-law |

These formats are applied to data words loaded into the SPORT transmit or receive data buffers. The first two data formats (00 and 01 values of SPORT\_CTL\_A.DTYPE) are applicable only when SPORT is configured as receiver; as when configured as transmitter, only the significant bits are transmitted (as per the field defined in control register). Therefore the transmit data buffers are not actually zero filled or sign extended.

The other two data formats enable the companding logic on the transmit/receive path. Companding (compressing or expanding) is the process of logarithmically encoding and decoding data to minimize the

number of bits to be sent. The processor's SPORTs support the two most widely used companding algorithms, A-law and  $\mu$ -law, which is performed according to the CCITT G.711 specification.

If selected, companding applies to both the enabled data channels. When enabled as SPORT transmitter, writes to transmit buffer causes its content compressed to eight bits (zero filled to the width of the transmit word) according to algorithm selected. Similarly, if configured in receive mode, the received 8-bits in the receive data buffers are expanded in right-justified, zero fill format as per the algorithm selected. If companding is enabled in multichannel mode, it is applied to all the active channels.

The compression for transmit data requires a minimum word length of 8 for proper function. If `SPORT_CTL_A.SLEN` is less than 7, then expansion may not work correctly. Also, if the data value is greater than 13-bit A-law or 14-bit  $\mu$ -law maximum, it is automatically compressed to the maximum value.

**NOTE:** The processor companding logic supports in-place companding feature. So, companding can be used as debug feature without enabling SPORT. See 'Companding as a Function' section for more details.

## Companding as a Function

Since the values in the transmit and receive buffers are actually companded in place, the companding hardware can be used without transmitting (or receiving) any data, for example during testing or debugging. For companding to execute properly, program the SPORT registers prior to loading data values into the SPORT buffers.

To compress data in place without transmitting use the following procedure.

1. Set the serial port as transmitter (`SPORT_CTL_A.SPTRAN = 1`) with both primary and secondary data channels disabled (`SPORT_CTL_A.SPENPRI = SPORT_CTL_A.SPENSEC = 0`).
2. Enable companding in the `SPORT_CTL_A.DTYPE` field.
3. Write a 32-bit data word to the transmit buffer
4. Wait for two system clock cycles. Any instruction not accessing the transmit buffer can be used to cause this delay. This allows the serial port companding hardware to reload the transmit buffer with the companded value.
5. Read the 8-bit compressed value from the transmit buffer.

To expand data in place, use the same sequence of operations with the receive buffer instead of the transmit buffer. When expanding data in this way, set the appropriate serial word length (`SPORT_CTL_A.SLEN`) bit.

## Transmit Path

The `SPORT_CTL_A.SPTRAN` control bit, when set, configures the SPORT in transmit mode. It then enables primary and/or secondary transmit paths, based on the `SPORT_CTL_A.SPENPRI` and `SPORT_CTL_A.SPENSEC` bit settings. Both data paths, primary and secondary, are separate but identical and include Transmit Data Buffer, optional companding logic and a Transmit Shift Register.

The data buffer on primary transmit path is known as Primary Transmit Data Buffer, or `SPORT_TXPRI_A`; while the one on secondary transmit path is known as Secondary Transmit Data Buffer, or `SPORT_TXSEC_A`. The transmit data buffer and output shift register forms a FIFO type of structure. When packing is disabled (`SPORT_CTL_A.PACK = 0`), serial port can hold as many as 3 data; while if packing is enabled (`SPORT_CTL_A.PACK = 1`), it can hold 2 packed data at any given time.

The data to be transmitted on primary and/or secondary channels is written to the `SPORT_TXPRI_A` and `SPORT_TXSEC_A` transmit data buffers respectively. The transmit data buffers can be accessed in core mode through peripheral bus or in DMA mode through DMA bus. The inactive data buffer must not be accessed. This data is optionally compressed in hardware according to selected algorithm and then automatically transferred to transmit shift register. The shift register, clocked by `SPORT_ACLK` signal, then serially shifts out this data on `SPORT_AD0` or `SPORT_AD1` pins, synchronously. If framing signal is used, the `SPORT_AFS` signal indicates the start of the serial word transmission.

When using DMA mode, a single DMA feeds the data buffers of the enabled channels (primary and or secondary). When using both channels, it is required to interleave the data of these channels properly.

*When SPORT is configured in non-multichannel mode as transmitter, the enabled SPORT data pins `SPORT_AD0` and/or `SPORT_AD1` are always driven. When a SPORT channel is enabled, data from Transmit Data Buffer is loaded into Transmit Shift register. The shift register then immediately latches the first bit of data (either LSB or MSB based on the `SPORT_CTL_A.LSBF` bit setting) and not with respect to frame sync. Similarly, if frame sync duration is greater than serial word length, then during inactive serial clock cycles (clock cycles after data transmission in the current frame), the data pins drives first bit of next word to be transmitted which loaded into shift register. This does not cause any problem at the receiver end, as it starts sampling the data pin only after detecting a valid frame sync. In multichannel mode, data pin always three-states during inactive channel slots.*

The serial port provides status of transmit data buffers and also error detection logic for transmit errors such as under-run. Please see the "Error Detection" section for more details.

When a serial port is configured in transmit mode, the receive paths (and hence the Receive Data Buffers and Receive Shift registers on those paths) are deactivated and do not respond to serial clock or frame sync signals. So, reading from an empty Receive Data Buffer may cause core to hang indefinitely.

## Receive Path

The `SPORT_CTL_A.SPTRAN` bit, when cleared, configures the SPORT in receive mode. It then enables primary and/or secondary receive paths, based on the `SPORT_CTL_A.SPENPRI` and `SPORT_CTL_A.SPENSEC` bit settings. Both data paths, primary and secondary, are separate but identical and include a Receive Shift Register, optional companding logic and a Receive Data Buffer.

The data buffer on primary receive path is known as Primary Receive Data Buffer, or `SPORT_RXPRI_A`; while the one on secondary receive path is known as Secondary receive Data Buffer, or `SPORT_RXSEC_A`. The receive paths act like a 3-word deep (32-bit) FIFO because they have two data registers plus an input shift register.



Upon enabling the serial port data channels, the input Shift register shifts in data bits on the SPORT\_ADO and/or SPORT\_AD1 pins, synchronous to the receive clock signal. If framing signal is used, the SPORT\_AFS signal indicates the beginning of the serial word being received. When an entire word is shifted in on the primary and secondary channels, the data is optionally expanded in hardware according to selected algorithm and then automatically transferred to SPORT\_RXPRI\_A and SPORT\_RXSEC\_A.

The Receive Data Buffers can be read in core mode through peripheral bus or in DMA mode through DMA bus. When DMA mode is used a single DMA reads the data buffers of enabled channels (primary and or secondary). When using both channels, it is required to de-interleave the data of these channels properly. The serial port provides the status of Receive Data buffers and also error detection logic for receive errors such as overflow. See the “Error Detection” section for more details.

When a serial port is configured in receive mode, the transmit paths (and the Transmit Data Buffers and Transmit Shift registers on those paths) are deactivated and do not respond to serial clock or frame sync signals. Therefore programs must not try to access them.

## Sampling Edge

The serial port uses two control signals to sample or drive the serial data.

1. Serial clock (SPORT\_ACLK) applies the bit clock for each serial data
2. Frame sync (SPORT\_AFS) divides the incoming data stream into frames.

These control signals can be internally generated or externally provided, determined by the SPORT\_CTL\_A.ICLK and SPORT\_CTL\_A.IFS bit settings.

Data and frame syncs can be sampled on the rising or falling edges of the serial port clock signals. The SPORT\_CTL\_A.CKRE bit controls the sampling edge. By default, when SPORT\_CTL\_A.CKRE = 0, the processor selects the falling edge of SPORT\_ACLK signal for sampling receive data and external frame sync. The receive data and frame sync are sampled on the rising edge of SPORT\_ACLK when SPORT\_CTL\_A.CKRE = 1.

Note that transmit data and internal frame sync signals are driven (change their state) on the serial clock edge that is not selected. By default, (SPORT\_CTL\_A.CKRE = 0) the SPORTs drive data and frame sync signals on the rising edge of the SPORT\_ACLK signal and drives on falling edge when SPORT\_CTL\_A.CKRE = 1.

Therefore transmit and receive functions of any two serial ports connected together should always select the same value for SPORT\_CTL\_A.CKRE so internally-generated signals are driven on one edge and received signals are sampled on the opposite edge.

The serial port which drives serial clock and frame sync is usually called as master while the receiver of clock and frame sync is referred as slave. The following figure shows the typical SPORT signals at two sides of serial communication for SPORT\_CTL\_A.CKRE = 0. The SPORT configured as Transmitter also drives the serial clock and Frame sync signals as a master device.

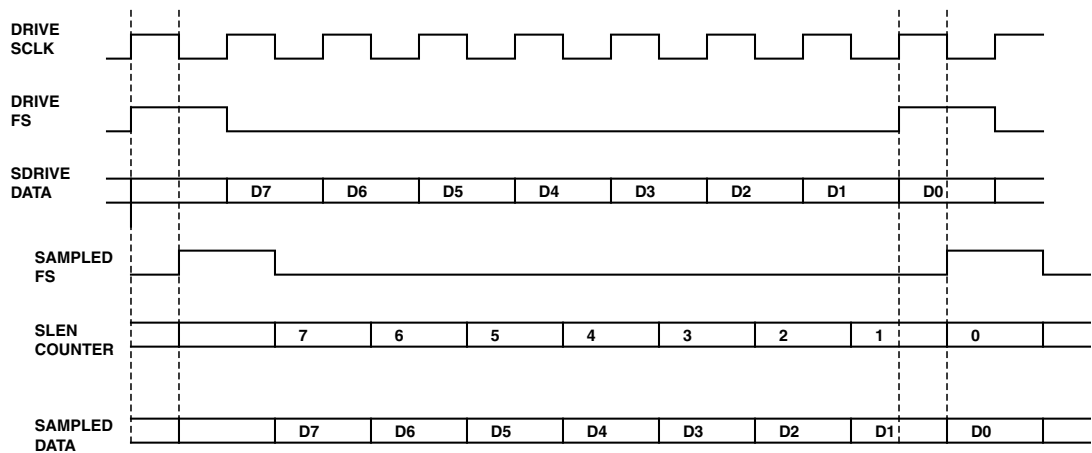


**Figure 26-3:** Frame Sync and Data Driven on Rising Edge

When slave samples the Frame Sync signal, the `SPORT_CTL_A.SLEN` word counter is reloaded to the maximum setting. Each `SPORT_ACLK` decrements the `SPORT_CTL_A.SLEN` counter until the full frame is received.

Therefore, if the transmitter drives the internal frame sync and data on the rising edge of serial clock, the falling edge should be used by receiver to sample the external frame sync and data, and vice versa.

Premature Frame Sync Error Detection

A SPORT framing signal is used to synchronize transmit or receive data. In external FS mode, any frame sync received when an active frame is in progress is called premature and is invalid.

As an enhancement to processor's serial port, if a premature frame sync is received, the `SPORT_ERR_A.FSERRSTAT` bit is flagged to indicate this framing error. An optional error interrupt can be generated for this event by setting `SPORT_ERR_A.FSERRMSK` bit.

This feature is applicable in all the operating modes of serial port.

**NOTE:** The `SPORT_ERR_A.FSERRSTAT` bit is not set in the presence of uncleared underflow/overflow errors.

In stereo or I<sup>2</sup>S mode, a premature frame sync may result in the SPORT receiving two consecutive left channels or two consecutive right channels and cause channel swapping. In the processor's serial port, swapping of channels due to a premature FS is avoided. If due to premature FS, one data gets corrupted, data will always be dropped in pairs to avoid channel swapping. The premature FS flagging in the error register will be done similarly.

As shown in the following figure, the frame sync error (which sets the error bit) is triggered when an early frame sync occurs during data transfer (transmission or reception) or for late frame sync if the period of the frame sync is smaller than the serial word length (`SPORT_CTL_A.SLEN`).

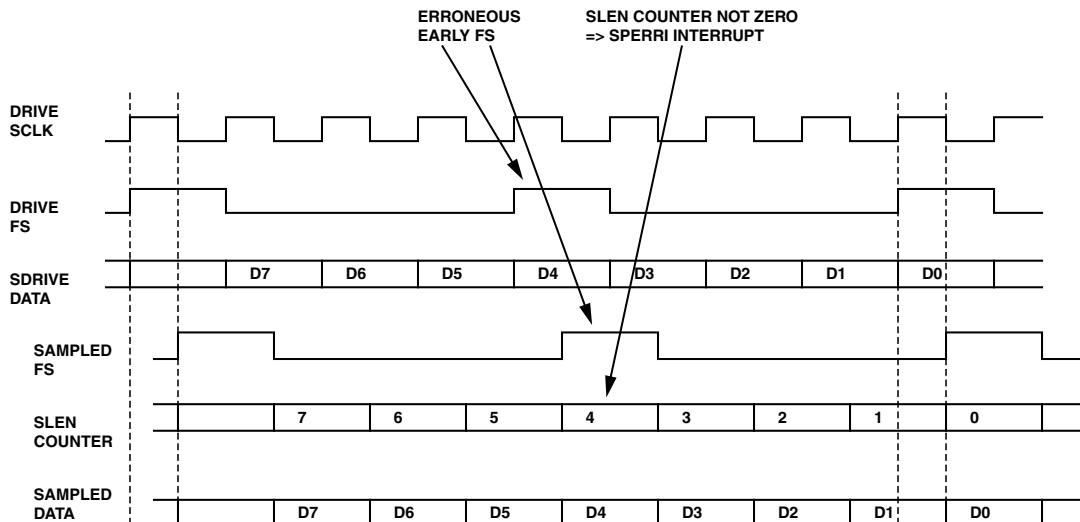


**Figure 26-4:** Frame Sync Error Detection

When a serial port is receiving or transmitting, its bit count is set to a word length (for example 32 bits). After each clock edge the bit count is decremented. After the word is received/ transmitted, the bit count reaches zero, and on next frame sync it is set to 32. When active transmission or reception is occurring, the bit count value is non-zero. When a frame sync with a bit count of non-zero is detected, a frame sync error occurs.

## Support for Edge-Detected and Level-Sensitive Frame Syncs

The frame sync signal of the SPORT module of earlier Blackfin processors is level-sensitive. Though the level sensitive nature of frame sync will work fine in a noise free environment; but if noise corrupts the signals coming into the SPORT, then there is a chance that the start of frame sync may be missed by the internal logic either because clock getting corrupted or frame sync itself. The frame sync will be sampled from the next clock edge onwards if the frame sync happens to last for more than a bit clock period.

The following figure describes a scenario when an external frame sync signal gets corrupted due to noise and is sampled incorrectly by the slave SPORT module. Consider a frame sync, driven on the rising edge of serial clock at  $t_A$  and expected to be sampled by slave SPORT at the falling edge of serial clock at  $t_B$ . But due to the noise, the first edge of the FS is not seen by the SPORT and samples the FS only at  $t_C$  as shown in the figure. Subsequently the word length counter runs for a period equal to the `SPORT_CTL_A.SLEN` field of the control register and expires at  $t_E$ , instead of correctly at  $t_D$ , receiving incorrect data. Further if a new frame sync edge has come at time  $t_D$ , in level sensitive mode, the SPORT samples this framing signal again only at time  $t_E$ . So, the frame sync sampling continues to be unaligned with the external data.

The enhanced SPORT module provides an option to configure the frame sync signal as edge-sensitive signal. Edge sensitive frame sync detection looks for an edge in an external frame sync for considering it as a valid framing signal. In active-high frame syncs, the rising edge of frame sync is considered valid; while in active-low frame syncs, the falling edge is considered as valid. This optional feature can be activated by programming `SPORT_CTL_A.FSED` (external Frame Sync Edge select) bit.

**NOTE:** `SPORT_CTL_A.FSED` is valid only in External frame sync mode. In internal frame sync mode, this bit is a don't care.

In the example discussed above, consider frame sync configured as edge-sensitive frame sync. In this case, frame sync will not be detected at  $t_E$  because the edge of framing signal had already come the previous cycle ( $t_D$ ) and there is no edge to detect at  $t_E$ . So the word length counter remains idle for this frame, ignoring the incorrect data, and resumes the operation correctly at  $t_F$  when a new edge detects.

This sets the `SPORT_ERR_A.FSERRSTAT` bit and optionally generates a premature FS error interrupt.

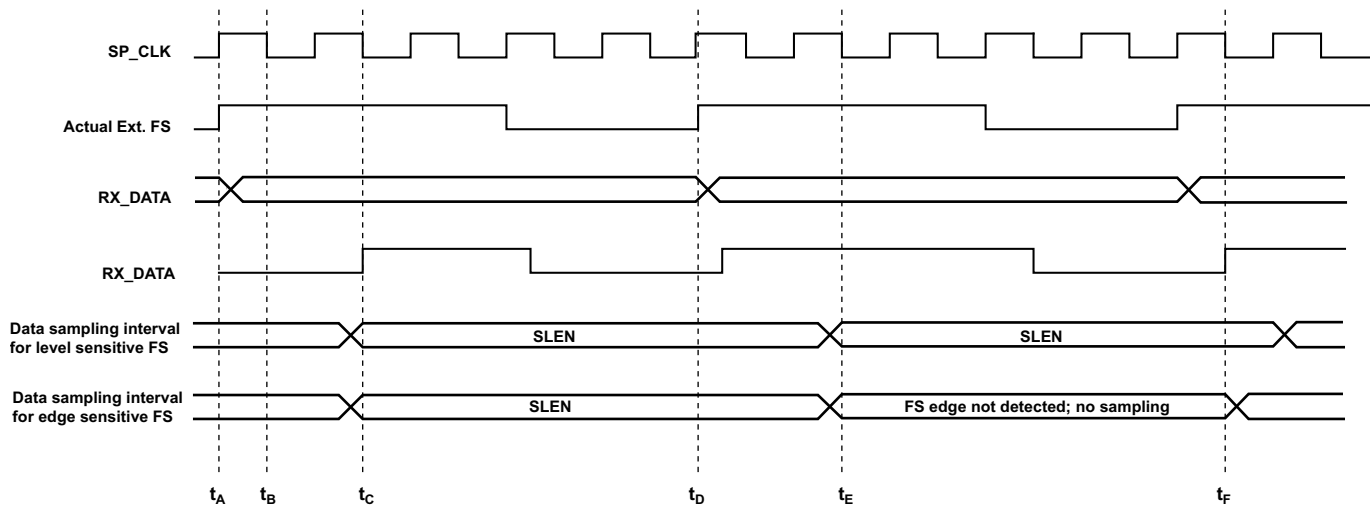


**Figure 26-5:** Level-Sensitive Frame Sync Vs. Edge Sensitive Frame Sync

Frame Sync edge detection is used by default for Stereo modes. MCM mode and DSP serial mode have an option to choose between edge detection and normal mode of FS detection.

**NOTE:** When the SPORT is enabled, an already active externally applied frame sync is not allowed to start operation. The SPORT waits for a valid state change from inactive to active for the external frame sync to consider it valid.

## Serial Word Length

The `SPORT_CTL_A.SLEN` field of serial port control register determines the word length of serial data to transmit and receive. Each SPORT half can independently handle word lengths up to 32 bits. The minimum allowable word length depends on operating mode selected. Words smaller than 32 bits are right-justified in the transmit or receive buffers to least significant bit (LSB) position. However, data can be shifted-in or out in MSB first or LSB first format according to `SPORT_CTL_A.LSBF` bit setting. Also, the received word can be sign-extended while storing it in processor memory.

The value of the `SPORT_CTL_A.SLEN` field can be calculated as:

$$\text{SLEN} = \text{Serial port word length} - 1$$

The range of valid word lengths in the operating modes of SPORT are as shown in the following table.

**Table 26-10:** Data Length Versus SPORT Operating Modes

| Mode                    | Serial Port word length (SLEN+1) |
|-------------------------|----------------------------------|
| Standard DSP Serial     | 4–32                             |
| I <sup>2</sup> S        | 5–32                             |
| Left-Justified          | 5–32                             |
| Right-Justified         | 5–32                             |
| Multichannel            | 5–32                             |
| Packed I <sup>2</sup> S | 5–32                             |

**NOTE:** If the companding feature is enabled on the data path, it limits the word length settings. See [Data Types and Companding](#) for more details about word lengths required for companding. If more than 32-bits per frame sync are required to transmit/receive, the multichannel mode can be used (by enabling more than one channel).

## Operating Modes

The SPORTs have six operating modes:

- Standard serial mode
- I<sup>2</sup>S mode
- Left-justified mode
- Right-Justified mode
- Multichannel mode
- Packed I<sup>2</sup>S mode

The SPORT halves within a SPORT can be independently configured in any of these operating modes, unless they are not coupled together using SPMUX logic. Each SPORT half has its own set of control and data registers and are programmed similarly.

The main control register of serial port, SPORT\_CTL\_A, controls the operating modes of the SPORT. The following table lists all the bits of the control register. The SPORT\_CTL\_A register is unique in that the bit function may change depending on the operating mode selected. It should be noted that many bits in the control registers, that control the function of the mode, are the same bit but have a different name depending on the operating mode. The bits common across operating mode columns (for example SPORT\_CTL\_A.SLEN) signifies that they function similarly across those operating modes. However, the bits

divided as per operating modes (for example `SPORT_CTL_A.LFS`) indicate different meaning depending on operating mode. Further, some bits are reserved depending on mode of operation (for example the `SPORT_CTL_A.FSR` bit is reserved in I<sup>2</sup>S, left-justified sample pair, packed I<sup>2</sup>S and in multichannel mode).

**NOTE:** When changing operating modes, clear the serial port control register before the new mode is written to the register.

**Table 26-11:** Control Bits comparison for different operating modes

| Bit (NAME)   | Standard Serial | I2S and<br>Left-Justified | Right-Justified | Multichannel | Packed I2S |
|--------------|-----------------|---------------------------|-----------------|--------------|------------|
| Control Bits |                 |                           |                 |              |            |
| 0 (SPENPRI)  | Yes             |                           |                 |              |            |
| 2–1 (DTYPE)  | Yes             | Reserved                  |                 | Yes          |            |
| 3 (LSBF)     | Yes             | Reserved                  |                 | Yes          |            |
| 8–4 (SLEN)   | Yes             |                           |                 |              |            |
| 9 (PACK)     | Yes             |                           |                 |              |            |
| 10 (ICLK)    | Yes             |                           |                 |              |            |
| 11 (OPMODE)  | Yes             |                           |                 |              |            |
| 12 (CKRE)    | Yes             | Reserved                  |                 | Yes          |            |
| 13 (FSR)     | Yes             | Reserved                  |                 |              |            |
| 14 (IFS)     | Yes             | Reserved                  |                 | Yes          |            |
| 15 (DIFS)    | Yes             |                           |                 | Reserved     |            |
| 16 (LFS)     | Yes             | Yes                       |                 | Yes          | Yes        |
| 17 (LAFS)    | Yes             |                           |                 | Reserved     |            |
| 18 (RJUST)   | Reserved        |                           | Yes             | Reserved     |            |
| 19 (FSED)    | Yes             | Reserved                  |                 | Yes          | Reserved   |
| 20 (TFIEN)   | Yes             |                           |                 |              |            |
| 21 (GCLKEN)  | Yes             |                           | Reserved        |              |            |

**Table 26-11:** Control Bits comparison for different operating modes (Continued)

| Bit (NAME)     | Standard Serial | I2S and<br>Left-Justified | Right-Justified | Multichannel | Packed I2S |
|----------------|-----------------|---------------------------|-----------------|--------------|------------|
| 24 (SPENSEC)   | Yes             |                           |                 |              |            |
| 25 (SPTRAN)    | Yes             |                           |                 |              |            |
| Status Bits    |                 |                           |                 |              |            |
| 26 (DERRSEC)   | Yes             |                           |                 |              |            |
| 28–27 (DXSSEC) | Yes             |                           |                 |              |            |
| 29 (DERRPRI)   | Yes             |                           |                 |              |            |
| 31–30 (DXSPRI) | Yes             |                           |                 |              |            |

## Mode Selection

The serial port operating mode is configured in the SPORT\_CTL\_A and the SPORT\_MCTL\_A registers. The following table provides values for each of the bits in the SPORT serial control registers that must be set in order to configure a specific SPORT operation mode. The shaded columns indicate that the bits come from different control registers.

**Table 26-12:** SPORT Operating Modes

| Operating Modes              | OPMODE (11) | LAFS (17) | RJUST (18) | Multichannel Control Register (SPORT_MCTL_A) |
|------------------------------|-------------|-----------|------------|----------------------------------------------|
| Standard DSP Serial          | 0           | Valid     | 0          | 0                                            |
| I2S                          | 1           | 0         | 0          | 0                                            |
| Left-Justified               | 1           | 1         | 0          | 0                                            |
| Right-Justified              | 1           | 1         | 1          | 0                                            |
| Multichannel                 | 0           | x         | 0          | 1                                            |
| Packed I <sup>2</sup> S mode | 1           | x         | 0          | 1                                            |

The following sections provide detailed information on each operating mode available using the serial ports.

## Standard Serial Mode

The SPORT can be configured in standard DSP serial mode by clearing the `SPORT_CTL_A.OPMODE` and `SPORT_MCTL_A.MCE` bits. The standard serial mode lets programs configure serial ports for use by a variety of serial devices such as serial data converters and audio codecs. In order to connect to these devices, a variety of clocking, framing, and data formatting options are available.

### Timing Control Bits

Several bits in the serial port control register enable and configure standard serial mode operation.

- **SLEN:** serial word length select (4-32 bits)
- **LSBF:** little endian Vs big endian serial bit format
- **ICLK:** Internal clock generation Vs external clock mode
- **CKRE:** sampling edge as rising edge Vs falling edge
- **IFS:** Internal frame sync generation Vs external FS mode
- **FSR:** framed mode Vs unframed mode
- **DIFS:** Data-dependent frame sync Vs data-independent frame sync
- **LFS:** active-high FS Vs active-low FS
- **LAFS:** early frame sync Vs late frame sync
- **PACK:** 16-bit to 32-bit packing enable Vs packing disable
- **GCLKEN:** normal free-running clock Vs Gated clock mode

### Clocking Options

In standard serial mode, the serial ports can either accept an external serial clock or generate it internally. The `SPORT_CTL_A.ICLK` bit determines the selection of these options. For internally-generated serial clocks (`SPORT_CTL_A.ICLK = 1`), the `SPORT_DIV_A.CLKDIV` field configures the serial clock rate from the system clock.



In addition, the serial clock edge can be selected for the sampling or driving serial data and/or frame sync. This selection is performed using the `SPORT_CTL_A.CKRE` bit.

- If `SPORT_CTL_A.CKRE = 0`, incoming data and/or frame sync signals are sampled with respect to falling edge of serial clock; while data and/or frame sync output signals are driven at the rising edge of clock.
- If `SPORT_CTL_A.CKRE = 1`, incoming data and/or frame sync signals are sampled with respect to rising edge of serial clock; while data and/or frame sync output signals are driven at the falling edge of clock.
- The `SPORT_CTL_A.GCLKEN` bit enables clock gating option, in which serial clock is active only during the valid data bits.

## Frame Sync Options

The following sections provide generic information about how frame sync signal is used by the serial port in an operating mode. Note that SPORT halves within a SPORT are independently configurable. Additional information about frame syncs and data sampling that applies to a specific operating mode can be found in [Operating Modes](#).

### Data-Dependent Versus Data-Independent Frame Sync

When a SPORT is configured as a transmitter (`SPORT_CTL_A.SPTRAN = 1`) and if data-independent frame sync select (`SPORT_CTL_A.DIFS`) bit = 0, then an internally-generated transmit frame sync is only output when a new data word has been loaded into the channel transmit buffer of the SPORT. In other words, frame sync signal generation and therefore data transmission is data-dependent. This mode of operation allows data to be transmitted only at specific times.

When SPORT is configured as receiver (`SPORT_CTL_A.SPTRAN = 0`) and if `SPORT_CTL_A.DIFS = 0`, then a receive frame sync signal is generated only when receive data buffer status is not full.

The data-independent frame sync mode allows the continuous generation of the framing signal, regardless of new data in the buffers. Setting `SPORT_CTL_A.DIFS` activates this mode. When `SPORT_CTL_A.DIFS = 1`, a transmit frame sync signal is generated regardless of the transmit data buffer status (if `SPORT_CTL_A.SPTRAN = 1`) or receive data buffer status (if `SPORT_CTL_A.SPTRAN = 0`).

Note that the SPORT DMA controller typically keeps the transmit buffer full. The application is responsible for filling the transmit buffers with data.

### Early Versus Late Frame Syncs

The frame sync signals can be early or late. Frame sync signals can occur during the first bit of each data word (late) or during the serial clock cycle immediately preceding the first bit (early). The `SPORT_CTL_A.LAFS` bit of the serial port control register configures this option.

By default, when `SPORT_CTL_A.LAFS` is cleared (=0), the frame sync signal is configured as early framing signal. This is the normal mode of operation. In this mode, the first bit of the transmit data word is available (and the first bit of the receive data word is latched) in the serial clock cycle after the frame sync is

asserted. The frame sync is not checked again until the entire word has been transmitted (or received). In multichannel operation, this is the case when the frame delay is one.

If data transmission is continuous in early framing mode (in other words, the last bit of each word is immediately followed by the first bit of the next word), then frame sync signal occurs during the last bit of each word. Internally-generated frame syncs are asserted for one clock cycle in early framing mode. This is not an error condition, so the `SPORT_ERR_A.FSERRSTAT` bit is not flagged.

When `SPORT_CTL_A.LAFS` is set ( $=1$ ), late frame syncs are configured; this is the alternate mode of operation. In this mode, the first bit of the transmit data word is available (and the first bit of the receive data word is latched) in the same serial clock cycle that the frame sync is asserted. In multichannel operation, this is the case when frame delay is zero. Receive data bits are latched by serial clock edges, but the frame sync signal is checked only during the first bit of each word. Internally-generated frame syncs remain asserted for the entire length of the data word in late framing mode. Externally-generated frame syncs are only checked during the first bit. They do not need to be asserted after that time period.

Therefore, for early framing, the frame sync precedes data by one cycle; for late framing, the frame sync is checked on the first bit only. The following figure illustrates the two modes of frame signal timing.

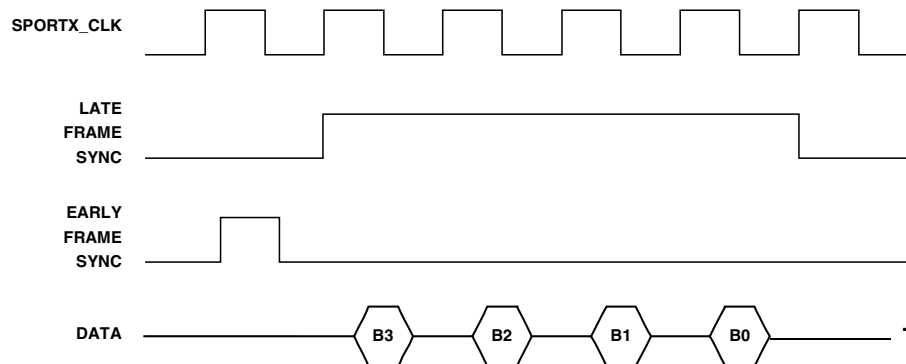


**Figure 26-6:** Normal Framing (Early Frame Sync) Versus Alternate Framing (Late Frame Sync)

### Framed Versus Unframed Frame Syncs

The use of frame sync signal is optional in serial port communications. The `SPORT_CTL_A.FSR` (frame sync required) bit determines whether framing signal is required.

When `SPORT_CTL_A.FSR` bit is set ( $=1$ ), a frame sync signal is required for every data word. To allow continuous transmission from the processor, each new data word must be loaded into the transmit buffer before the previous word is shifted out and transmitted.

When `SPORT_CTL_A.FSR` is cleared ( $=0$ ), the corresponding frame sync signal is not required. A single frame sync is required to initiate communications but it is ignored after the first bit is transferred. Data words are then transferred continuously in what is referred to as an unframed mode. Unframed mode is appropriate for continuous reception. The following figure shows the framed vs unframed mode of serial port operation.

**NOTE:** When DMA is enabled in a mode where frame syncs are not required, DMA requests may be held off by chaining or may not be serviced frequently enough to guarantee continuous unframed data flow. Monitor status bits or check for a SPORT Error interrupt to detect underflow or overflow of data.

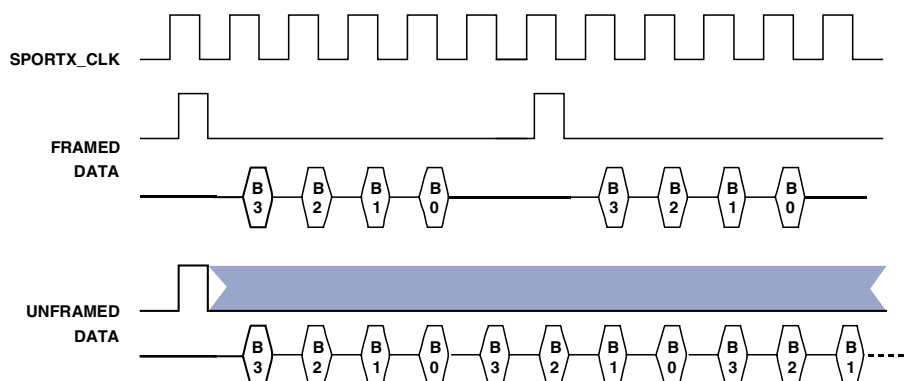


**Figure 26-7:** Framed Versus Unframed Data Stream

## Logic Level

The framing signals may be active high or active low. The `SPORT_CTL_A.LFS` bit selects the logic level of the frame sync signals.

- When `SPORT_CTL_A.LFS = 0`, the corresponding frame sync signal is active high.
- When `SPORT_CTL_A.LFS = 1`, the corresponding frame sync signal is active low.

Active high is the default polarity of frame sync signal.

## Stereo Modes

The processor serial port support three widely used stereo modes, which are I<sup>2</sup>S mode, Left-Justified mode and right-Justified mode. In these modes, the serial data stream consists of left and right channels. These modes are described in the following sections.

## Channel Order First

The active low frame sync (`SPORT_CTL_A.LFS`) bit, which determines the polarity of frame sync level in DSP serial mode/multichannel mode, holds different meaning for stereo modes of SPORT operation. For left-justified, I<sup>2</sup>S and packed I<sup>2</sup>S modes, the following table demonstrates which word is transmitted or receive first depending on the `SPORT_CTL_A.LFS` bit setting.

**Table 26-13:** Channel Order First Bit Settings

| OPMODE                  | LFS=0 (Left Channel First, Default) | LFS=1 (Right Channel First)   |
|-------------------------|-------------------------------------|-------------------------------|
| Left-Justified          | Data first after rising edge        | Data first after falling edge |
| I <sup>2</sup> S        | Data first after falling edge       | Data first after rising edge  |
| Packed I <sup>2</sup> S | Data first after rising edge        | Data first after falling edge |

## I<sup>2</sup>S Mode

I<sup>2</sup>S mode is a very commonly used stereo mode where in each frame sync cycle two samples of data are transmitted/received—one sample on the low segment of the frame sync, known as left channel; while the other sample on the high segment of the frame sync, known as right channel.

The SPORT can be configured in I<sup>2</sup>S mode by setting `SPORT_CTL_A.OPMODE = 1`, `SPORT_CTL_A.LFS = 0` and `SPORT_MCTL_A.MCE = 0`.

## Protocol Configuration Options

Several bits in the `SPORT_CTL_A` control register enable and configure I<sup>2</sup>S mode of operation:

- **SLen:** serial word length select. For I<sup>2</sup>S mode, the range of allowable word length is 5-32 bits.
- **LSBF:** little endian or big endian serial bit format. For I<sup>2</sup>S mode, serial data should be in big endian format (MSB bit is transmitted/received first). But, it can be changed depending on the user's preference when emulating similar non-standard protocol
- **ICLK:** Internal bit clock generation or external bit clock mode
- **IFS:** Internal frame sync generation or external FS mode. In I<sup>2</sup>S mode, master serial port is the one which generates bit clock and L/R clock (frame sync) internally for the serial communication. The serial port which accepts these clocking signals is known as slave. So, in standard I<sup>2</sup>S mode, the `SPORT_CTL_A.IFS` bit should depend (equal to) on `SPORT_CTL_A.ICLK` bit setting. However, as an enhancement, the `SPORT_CTL_A.IFS` bit setting may be changed depending on the application when emulating a non-standard protocol.
- **LFS:** left channel first or right channel first. This bit setting may be changed depending on user's preference to sample right channel first (LFS = 0) or left channel first (LFS = 1).
- **CKRE:** sampling edge as rising edge or falling edge.
- **PACK:** 16-bit to 32-bit packing enable or packing disable.

## Serial Clock and Frame Sync Rates

The serial bit clock rate for internal clocks can be set using a `SPORT_DIV_A.CLKDIV` bit field; while the L/R clock rate for internal frame sync can be set using the `SPORT_DIV_A.FSDIV` bit field in the same register, depending on `SPORT_CTL_A.ICLK` and `SPORT_CTL_A.IFS` bit settings.

The following figure shows the timing in I<sup>2</sup>S mode. Note that in I<sup>2</sup>S mode, the data is delayed by one SCLK cycle and the operation transfer starts on the left channel first.

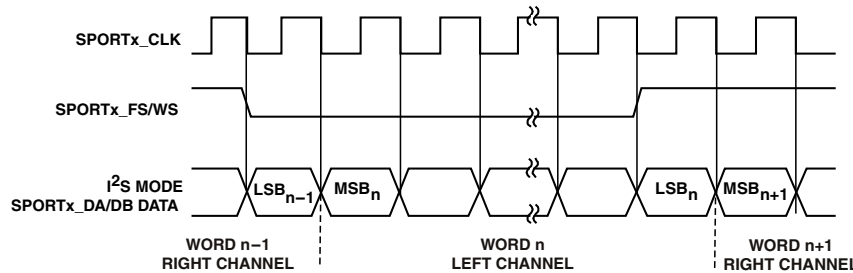


**Figure 26-8:** Word Select Timing in I2S Mode

Unlike serial port of SHARC processor, the serial port of ADSP-BF60x does not generate a frame sync (L/R clock) edge after the transmission of last word in the DMA (same as behavior in the earlier Blackfin processors). Standard I<sup>2</sup>S receivers look for the edge to latch and read data. Therefore I<sup>2</sup>S slave receivers connected to Blackfin SPORT may not be able to latch the last word of the TX DMA.

## Left-Justified Mode

Left-justified mode is a stereo mode where in each frame sync cycle two samples of data are transmitted/received—one sample on the high segment of the frame sync, known as left channel; while the other sample on the low segment of the frame sync, known as right channel.

This operating mode is simply a subset of the I2S mode. The SPORT can be configured in Left-Justified mode by setting the `SPORT_CTL_A.OPMODE` and `SPORT_CTL_A.LAFS` bits and clearing the `SPORT_MCTL_A.MCE` bit.

## Protocol Configuration Options

Several bits in the `SPORT_CTL_A` control register enable and configure the left-justified mode of operation:

- **SLEN:** serial word length select. For left-justified mode, the range of allowable word length is 5-32 bits.
- **LSBF:** little endian or big endian serial bit format. For left-justified mode, serial data should be in big endian format (MSB bit is transmitted/received first). But, it can be changed depending on the user's preference when emulating similar non-standard protocol
- **ICLK:** Internal bit clock generation or external bit clock mode
- **IFS:** Internal frame sync generation or external FS mode. In left-justified mode, master serial port is the one which generates bit clock and L/R clock (frame sync) internally for the serial communication. The serial port which accepts these clocking signals is known as slave. So, in standard left-justified mode,

the `SPORT_CTL_A.IFS` bit should depend (equal to) on `SPORT_CTL_A.ICLK` bit setting. However, as an enhancement, the `SPORT_CTL_A.IFS` bit setting may be changed depending on the application when emulating a non-standard protocol.

- **LFS:** left channel first or right channel first. In standard left-justified mode, left channel is sampled first (corresponds to `SPORT_CTL_A.LFS = 0`). However, `SPORT_CTL_A.LFS` bit setting may be changed depending on user's preference to sample right channel first (first data after rising edge of L/R clock).
- **CKRE:** sampling edge as rising edge or falling edge.
- **PACK:** 16-bit to 32-bit packing enable or packing disable.

## Serial Clock and Frame Sync Rates

The serial bit clock rate for internal clocks can be set using a `SPORT_DIV_A.CLKDIV` field; while the L/R clock rate for internal frame sync can be set using `SPORT_DIV_A.FSDIV` field, depending on `SPORT_CTL_A.ICLK` and `SPORT_CTL_A.IFS` bit settings. The following figure shows the serial port timing in left-justified mode (it is shown in MSB bit first format, but LSB bit first format is also possible). As shown, the first bit of a word is transmitted/received in the same clock cycle as the word select (`SPORT_FS`) signal changes.

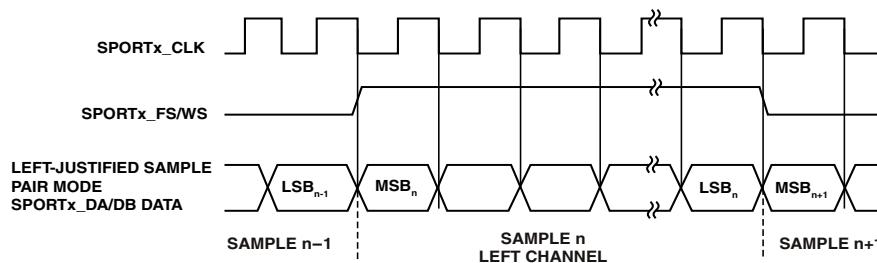


**Figure 26-9:** Word Select Timing in Left-Justified Mode

## Right-Justified Mode

Right-justified mode is a standard commonly available in most of the SPORT compatible devices such as ADC and DACs. Right-justified mode requires that the design align the data to the end of the frame sync. While the SPORTs in earlier Blackfin processors aligned the serial data with the start of the frame sync (used with left-justified or  $I^2S$  mode) the ADSP-BF60x has a `SPORT_CTL_A.RJUST` bit that aligns to the end of the frame sync.

The following figure shows the SPORT timing in right-justified mode. As shown, the transmitter aligns the data to be transmitted such that the last bit of the serial word is sent in the last clock cycle of the word select (frame sync) signal marking the channels. The timing seems similar to left-justified mode (where the transmitter sends the MSB bit of the serial word in the same clock cycle as the word select signal changes), but data is shifted such that it aligns to end of the channel.

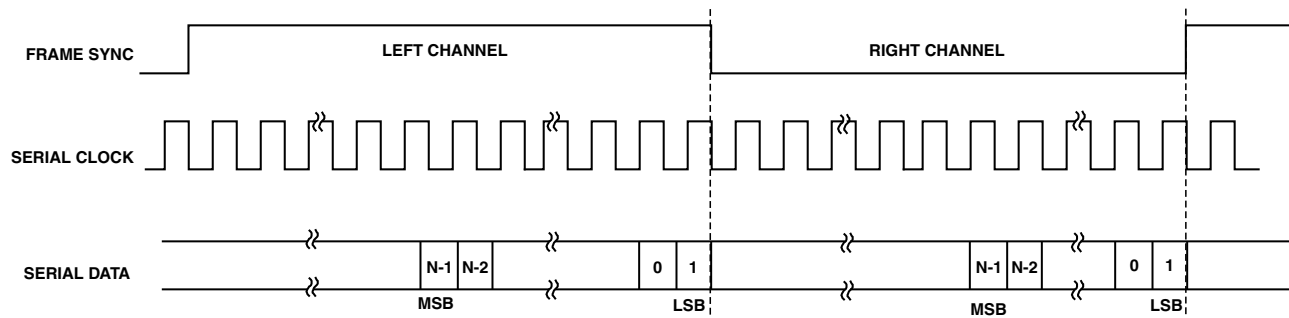


**Figure 26-10:** Word Select Timing in Right-Justified Mode

**NOTE:** For some SPORT compatible ADC or DACs (for example the AD1871) right-justified mode is limited to some commonly used ratios such as 64 FS and 128 FS as the bit clock frequency (where FS is the sampling frequency of ADC/DACs, known as L/R clock or the frame sync of the SPORT). The right-justified mode implemented in the ADSP-BF60x SPORT supports up to 128 FS.

As an illustration, consider the SPORT timing for right-justified mode, as shown in the figure below. If the L/R clock runs at the FS rate and the SPORT clock at the 64 FS rate, the frame sync width (either channel) is limited to 32 SPORT clock periods or 32 bits per channel. If the data is confined to 24 bits, the SPORT introduces a  $32 - 24 = 8$  bit clock delay before it starts to transmit/capture data.

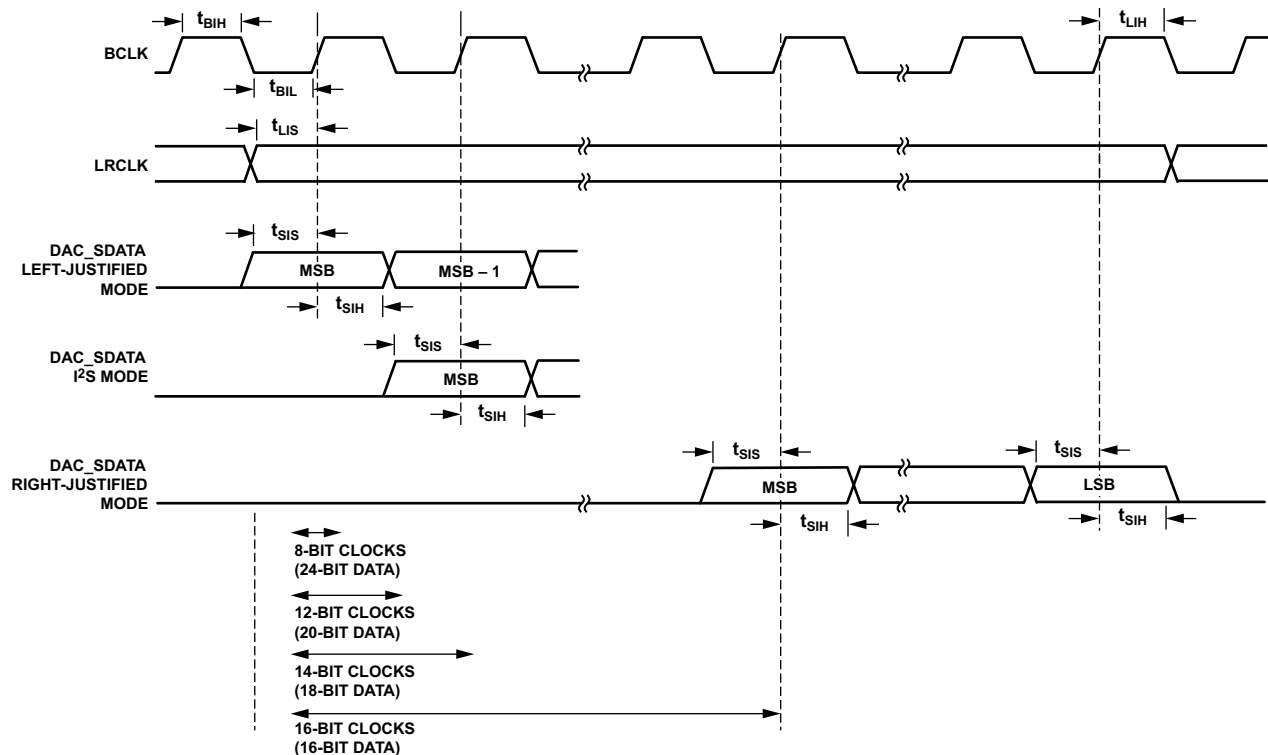


**Figure 26-11:** Timing Comparison Between Different Stereo Modes

Similarly, if 128 FS bit clock frequency is to be supported, then the frame sync width becomes 64 serial clock periods (bits) per channel. In this case, the delay can be a maximum of  $(64 - \text{minimum serial data})$

length in right-justified mode) = 59 bits (the minimum `SPORT_CTL_A.SLEN` setting is 4). This implies that a 6-bit counter is needed to set this delay.

Therefore, using this counter in right-justified mode, the starting point of the first bit is delayed so that the serial data is aligned properly with the end of the channel. A 6-bit counter is added for this purpose in the stereo mode. This counter is programmed by writing into the `SPORT_MCTL_A` 16-21 bit field. Note that these bits are used to configure the window offset size in multichannel mode. But since stereo serial mode and multichannel mode are mutually exclusive, the separation of role of this field in each mode is clearly defined and implemented. The software has to configure this register with the appropriate delay keeping in mind the word length (`SPORT_CTL_A.SLEN`) and the number of bit clocks in one channel (left/right).

## Timing Control Bits

The following bits in the `SPORT_CTL_A` register enable and configure right-justified mode.

- **SLEN:** serial word length select. For right-justified mode, the range of allowable word length is 5-32 bits.
- **LSBF:** little endian or big endian serial bit format. For right-justified mode, serial data should be in big endian format (MSB bit transmitted/received first). But, it can be changed depending on the application when using non-standard protocol.
- **ICLK:** Internal bit clock generation or external bit clock mode
- **IFS:** Internal frame sync generation or external FS mode.
- **LFS:** left channel first or right channel first. In standard right-justified mode, the left channel is sampled first (corresponds to `SPORT_CTL_A.LFS = 0`). However, the `SPORT_CTL_A.LFS` bit setting may be changed depending on the application to sample the right channel first (first data after falling edge of L/R clock).
- **CKRE:** sampling edge as rising edge or falling edge.
- **PACK:** 16-bit to 32-bit packing enable or packing disable.
- **MCTL16-21:** 6-bit counter depends on `SPORT_CTL_A.SLEN` and number of bit clocks in a channel.

## Serial Clock and Frame Sync Rates

The serial bit clock rate for internal clocks can be set using the `SPORT_DIV_A.CLKDIV`; while the L/R clock rate for internal frame sync can be set using the `SPORT_DIV_A.FSDIV` bit field in the same register, depending on `SPORT_CTL_A.ICLK` and `SPORT_CTL_A.IFS` bit settings.

## Multichannel Mode

The processor's SPORTs offer a multichannel mode of operation, which allows the SPORT to communicate in a time division multiplexed (TDM) serial system. In multichannel communications, each data word



of the serial bit stream occupies a separate channel. Each word belongs to the next consecutive channel. For example, a 24-word block of data contains one word for each of the 24 channels.

The multichannel mode of SPORT can be selected by setting `SPORT_CTL_A.OPMODE = 0` and `SPORT_MCTL_A.MCE = 1`.

Up to 128 channels are available for transmitting or receiving. The SPORT can automatically select some words for particular channels while ignoring others. In other words, each SPORT can receive or transmit data selectively from any of the 128 channels. These 128 channels can be any 128 out of the 1024 total channels in the system. The SPORT can do any of the following on each channel:

- Transmit data (`SPORT_CTL_A.SPTRAN = 1`)
- Receive data (`SPORT_CTL_A.SPTRAN = 0`)
- Do nothing during inactive channels

Optionally, data companding and DMA transfers can be used in multichannel mode on both primary and secondary data lines.

The SPORT multichannel select registers (`SPORT_CS0_A`) must be programmed before enabling SPORT operation for multichannel mode. This is especially important in DMA data unpacked mode, since the SPORT data buffers begin operation immediately after the SPORT data lines are enabled. The `SPORT_MCTL_A.MCE` must also be enabled prior to enabling SPORT operation.

Multichannel mode operates completely independently and each SPORT uses its own serial clock and frame sync signal either internally generated or externally provided.

## Protocol Configuration Options

The following bits in the `SPORT_CTL_A` and `SPORT_MCTL_A` registers enable and configure multichannel mode.

- SLEN: serial word length select
- LSBF: little endian or big endian serial bit format
- ICLK: Internal clock generation or external clock mode
- CKRE: sampling edge as rising edge or falling edge
- IFS: Internal frame sync generation or external FS mode
- LFS: active-high FS or active-low FS
- PACK: 16-bit to 32-bit packing enable or packing disable
- MFD: Multichannel frame delay
- WSIZE: Number of multichannel channels
- WOFFSET: window offset size

- MCPDE: Multichannel DMA packing enable

## Clocking Options

In multichannel mode, the SPORTs can either accept an external serial clock or generate it internally. The `SPORT_CTL_A.ICLK` bit determines the selection of these options. For internally-generated serial clock (`SPORT_CTL_A.ICLK = 1`), of the `SPORT_DIV_A.CLKDIV` bit field configures the serial clock from the system clock.

In addition, the serial clock edge can be selected for the sampling or driving serial data and/or frame sync. This selection is performed using the `SPORT_CTL_A.CKRE` bit.

- If `SPORT_CTL_A.CKRE = 0`, incoming data and/or frame sync signals are sampled with respect to falling edge of serial clock; while data and/or frame sync output signals are driven at the rising edge of clock.
- If `SPORT_CTL_A.CKRE = 1`, incoming data and/or frame sync signals are sampled with respect to rising edge of serial clock; while data and/or frame sync output signals are driven at the falling edge of clock.

## Frame Sync Options

The frame sync signal synchronizes the channels and restarts each multichannel sequence. The `SPORT_FS` signal initiates the start of the channel 0 data word. The frame sync period in multichannel is defined as:

$$\text{FS period} = [(\text{SPORT\_CTL\_A.SLEN} + 1) \times \text{number of channels}] - 1$$

The frame sync can be configured in master or slave mode based on the setting of the `SPORT_CTL_A.IFS` bit and its logic level can be changed using the `SPORT_CTL_A.LFS` bit.

In multichannel mode, frame sync timing is similar to late frame mode (though the `SPORT_CTL_A.LAFS` bit is reserved in this mode)—the first bit of the transmit data word is available and the first bit of the receive data word is sampled in the same serial clock cycle that the frame sync is asserted, provided that multichannel frame delay (`SPORT_MCTL_A.MFD`) is set to 0.

The frame sync signal is used for the block or frame start reference, after which the word transfers are performed continuously with no further frame sync signals required during the ongoing frame for different channels. Therefore, internally generated frame syncs are always data independent (`SPORT_CTL_A.DIFS` bit is reserved).

## Transmit Data Valid (TDV)

Each serial port has its own Transmit Data Valid signal (`SPORT_ATDV`) which is active during the transmission of enabled words. Because the serial port signals are three-stated when the time slot is not active, the `SPORT_ATDV` signal specifies if the SPORT data is being driven by the processor. It serves as an output-enabled signal for the data transmit pin. After the transmit data buffer is loaded, transmission begins and the `SPORT_TDV` signal is asserted.

The polarity of this Transmit Data Valid signal is always active high in that `SPORT_ATDV` is asserted high when a data is transmitted during the active channel slot of serial port.

The following figure shows an example of timing for a multichannel transfer having following characteristics.

- The half SPORT pair of SPORT0, A and B, is configured as a transmitter and receiver respectively; while half SPORT A of SPORT1 is configured as transmitter.
- The serial clock and frame sync signals are input to all of these HSPORTs.
- Only primary channels of these SPORTs are enabled (0).
- Multichannel is configured to 8 channels.
- SPORT0\_A drives data (on its primary data line) during slot 1–0 which asserts SPORT0\_ATDV for 2 slots.
- SPORT1\_A drives data (on its primary data line) during slot 3–2 which asserts SPORT1\_ATDV for 2 slots.
- SPORT0\_B receives data (from its primary data line) during slot 3–0.

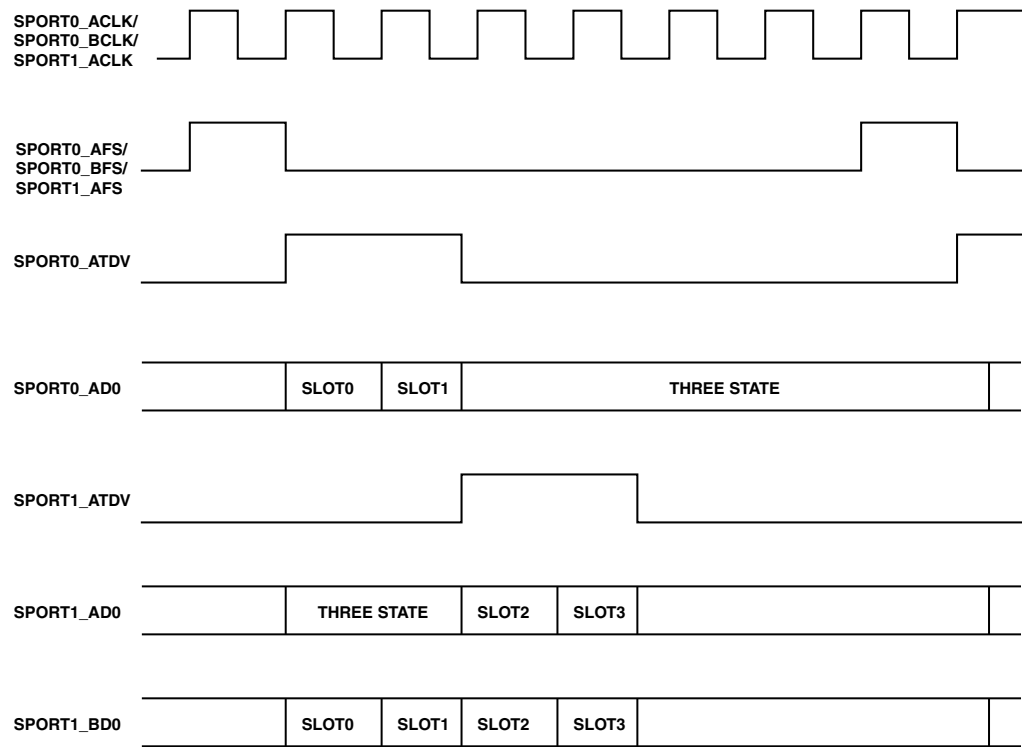


Figure 26-12: Multichannel Timing

Active Channel Selection Registers (SPORT\_CS0\_A)

In multichannel mode, SPORT supports up to 128 channels for transmitting or receiving. It can receive or transmit data selectively from any of the 128 channels. Specific channels can be individually enabled or disabled, using multichannel selection registers (CSx), to select the words that are transmitted or received

during multichannel communications. Data words from the enabled channels are transmitted or received, while disabled channel words are three-stated or ignored.

Each of the four multichannel selection registers is 32 bits in length. Therefore these registers provide channel selection for 128 (0 to 127) channels. Setting a bit enables that channel so that the serial port selects its word from the multiple-word block of data (for either receive or transmit). The 128 channels are sequentially numbered from bit 0 in the CS0 register to bit 31 of SPORT\_CS3\_A register. As an example setting bit 13 of the SPORT\_CS1\_A register enables channel number 45 ( $31+13+1$ ); similarly setting bit 5 of the SPORT\_CS3\_A register enables channel number 101 ( $31+32+32+5+1$ ).

## **Multichannel Frame Delay (MFD)**

The 4-bit multichannel frame delay (SPORT\_MCTL\_A.MFD) field in the multichannel control registers (MCTL\_x) specifies a delay between the frame sync pulse and the first data bit in frame. The value of SPORT\_MCTL\_A.MFD is the number of serial clock cycles of the delay. This multichannel frame delay allows the processor to work with different types of telephony interface devices.

A value of zero for SPORT\_MCTL\_A.MFD causes the frame sync to be concurrent with the first data bit. The maximum value allowed for SPORT\_MCTL\_A.MFD is 15. If SPORT\_MCTL\_A.MFD>0, a new frame sync may occur during the last channels of a previous frame, which is considered as valid frame sync signal. *If more than 15 bits frame delay is required, the Window Offset field may be used to delay the start of channel 0.*

## **Number of Multichannel Slots (WSIZE)**

Select the number of channels used in multichannel operation by using the 7-bit SPORT\_MCTL\_A.WSIZE field in the multichannel control register. Set SPORT\_MCTL\_A.WSIZE to the actual number of channels minus one ( $\text{SPORT\_MCTL\_A.WSIZE} = \text{Number of channels} - 1$ ). So, the granularity of number of channels selected is 1.

A 10-bit field in the multichannel mode status register, SPORT\_MSTAT\_A, holds the channel number which is being serviced in the multichannel operation.

## **Window Offset (WOFFSET)**

The window offset (SPORT\_MCTL\_A.WOFFSET25-16) field of the SPORT\_MCTL\_A register specifies where in the 1024-channel range to place the start of the active window. A value of 0 specifies no offset and 896 ( $1024 - 128$ ) is the largest value that allows using all 128 channels.

As an example, a program could define an active window with 7 multichannel slots ( $\text{SPORT\_MCTL\_A.WSIZE} = 8$ ) and an offset of 93 ( $\text{SPORT\_MCTL\_A.WOFFSET} = 93$ ). This 8-channel window then resides in the range from 93 to 100.

Neither the window offset nor the number of multichannel slots (SPORT\_MCTL\_A.WSIZE) can be changed while the SPORT is enabled. If the combination of the window size and the window offset would place any portion of the window outside of the range of the channel counter, none of the out-of-range channels in the frame are enabled.

## Companding Selection

Like all other operating modes, companding logic can be applied to serial data. In transmit mode, compression logic is applied to the data to be transmitted; while in receive mode, expansion logic can be applied to received data. The two widely used companding algorithms, A-law and  $\mu$ -law can be applied by configuring `SPORT_CTL_A.DTYPE` field of the control register.

If companding is enabled, the companding algorithm is applied to both the data paths. In multichannel mode, companding can be applied to either all or none of the enabled channels, (companding cannot be selected on a per-channel basis).

## Multichannel DMA Data Packing (MCPDE)

Multichannel DMA data packing and unpacking are specified with the `SPORT_MCTL_A.MCPDE` bit setting.

If the bits are set, indicating that data is packed, the SPORT expects the data contained by the DMA buffer corresponds only to the enabled SPORT channels. For example, if an MCM frame contains 10 enabled channels, the SPORT expects the DMA buffer to contain 10 consecutive words for each frame.

If the bits are cleared (the default, indicating that data is not packed), the SPORT expects the DMA buffer to have a word for each of the channels in the active window, whether enabled or not, so the DMA buffer size must be equal to the size of the window. For example, if only channel number 1 and 10 are enabled, then DMA buffer size would have to be 10 words (unless the secondary side is enabled). The data to be transmitted or received would be placed at addresses 1 and 10 of the buffer, and the rest of the words in the DMA buffer would be ignored.

## Multichannel Frame

A multichannel frame contains more than one channel, as specified by the `SPORT_MCTL_A.WSIZE` field and window offset field of multichannel control register. A complete multichannel frame consists of 1–1024 channels, starting with channel 0. The particular channels of the multichannel frame that are selected for the SPORT are a combination of the window offset, the window size, and the multichannel select registers.

The following figure illustrates the relationship between different parameters of multichannel timings. Frame length is set by frame sync divider or external frame sync period

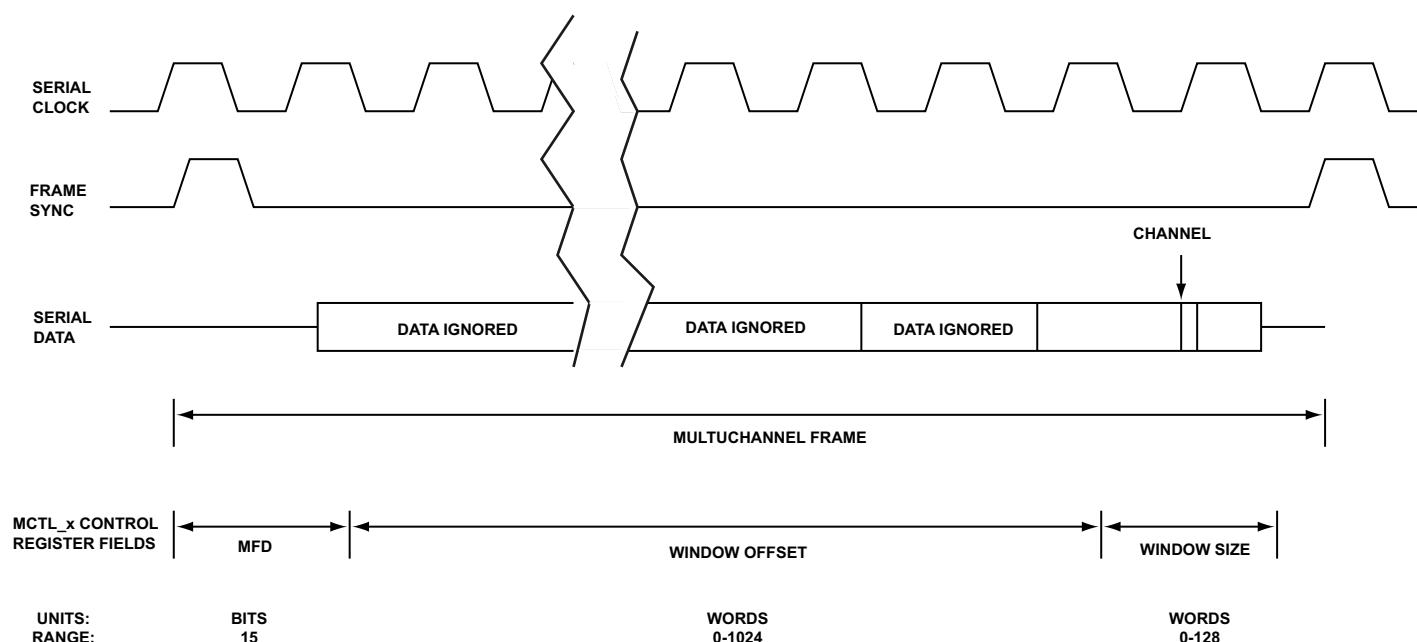


**Figure 26-13:** Relationships for Multichannel Parameter

## Packed I<sup>2</sup>S Mode

A packed I<sup>2</sup>S mode is available in the SPORT and used for audio codec communications using multiples channels. This mode allows applications to send more than the standard 32 bits per channel available through standard I<sup>2</sup>S mode. Packed mode is implemented using standard multichannel mode (and is therefore programmed similarly to multichannel mode).

The SPORT can be configured in packed I<sup>2</sup>S mode by setting the `SPORT_CTL_A.OPMODE` bit and the `SPORT_MCTL_A.MCE` bit.

Similar to multichannel mode, packed I<sup>2</sup>S mode also supports the maximum of 128 channels as the maximum of (128 x 32) bits per left or right channel.

As shown in the following figure, packed waveforms are the same as the wave forms used in multichannel mode, except that the frame sync is toggled for every frame, and therefore emulates I<sup>2</sup>S mode. So it is a hybrid between multichannel and I<sup>2</sup>S mode.

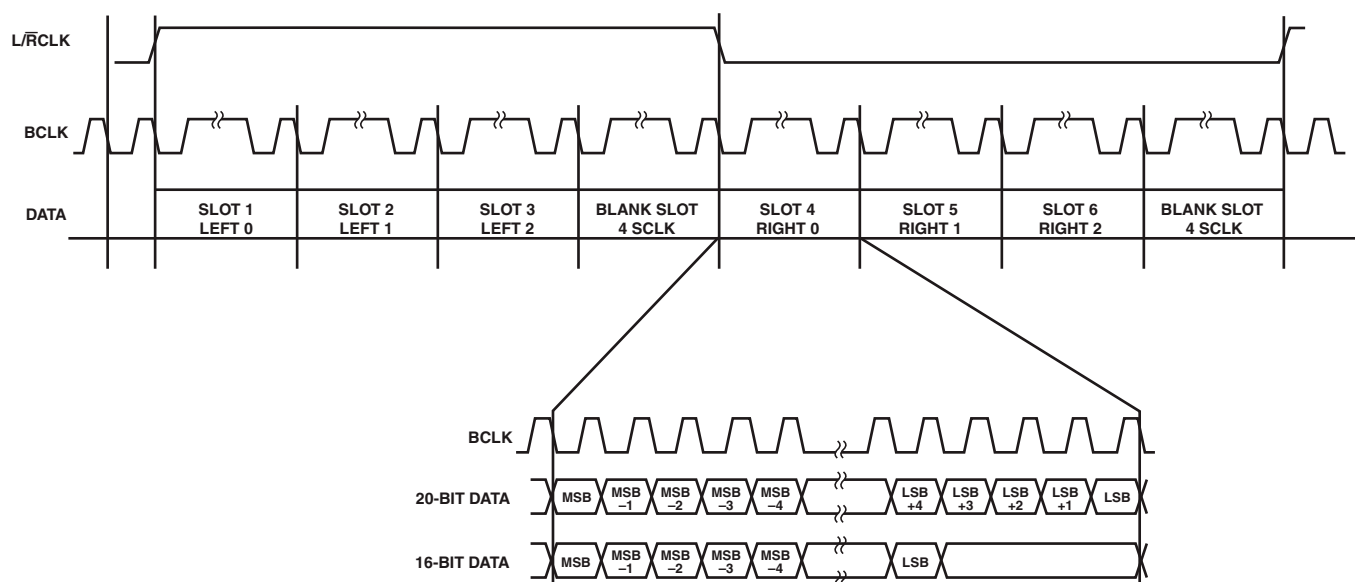


**Figure 26-14:** Packed I<sup>2</sup>S Mode 128 Operation

## Protocol Configuration Options

Several bits in the SPORT\_CTL\_A and SPORT\_MCTL\_A registers enable and configure packed I<sup>2</sup>S mode.

- SLEN: serial word length select
- LSBF: little endian or big endian serial bit format
- ICLK: Internal clock generation or external clock mode
- CKRE: sampling edge as rising edge or falling edge
- IFS: Internal frame sync generation or external FS mode
- LFS: left-channel first or right channel first
- PACK: 16-bit to 32-bit packing enable or packing disable
- MFD: Multichannel frame delay
- WSIZE: Number of multichannel channels
- WOFFSET: window offset size
- MCPDE: Multichannel DMA packing enable

## Clocking Options

In packed mode, the serial ports can either accept an external serial clock or generate it internally. The SPORT\_CTL\_A.ICLK register determines the selection of these options. For internally-generated serial

clock (`SPORT_CTL_A.ICLK = 1`), the `SPORT_DIV_A.CLKDIV` bit field configures the serial clock rate from the system clock.

The programs can also select the serial clock edge that is used for sampling or driving serial data and/or frame syncs. This selection is performed using the `SPORT_CTL_A.CKRE` bit.

## Frame Sync Options

The frame sync period in packed mode is defined as:

$$\text{FS period} = [\text{SPORT\_CTL\_A.SLEN} + 1] \times \text{number of channels} - 1.$$

The frame sync can be configured in master or slave mode depending on the `SPORT_CTL_A.IFS` bit setting of serial port control register. Moreover the logic level can be changed with the `SPORT_CTL_A.LFS` bit setting.

## Gated Clock Mode

Some of the ADC/DACs support the SPI compatible protocol for the interface. To communicate with such ADC/DACs, the serial port must support the gated clock mode of operation, in which the data valid information is embedded into the clock. Therefore, in gated clock mode of, the clock should be active only when active data is being transmitted or received.

The processor features the gated clock function. The `SPORT_CTL_A.GCLKEN` (Gated clock mode select) control bit, is used to configure the serial port in gated clock mode. To enable gated clock mode of operation, SPORT must be programmed to comply with the following requirements.

- Gated clock mode feature is available in standard serial mode, left-justified and I<sup>2</sup>S mode of operation only.

[Note that among the stereo modes, right justified mode cannot be operated in gated clock mode because during the inactive period in a frame (the period between the leading edge of the frame sync and the first active data bit), there is a delay counter running inside the serial port. The counter operates on the serial clock and any interruption in clock makes the counter go out of sync].

- Gated clock mode has the following valid settings for other control bits.
  - Both serial clock and frame sync signals generated internally
  - Both serial clock and frame sync signals provided externally
  - Frame sync not required' mode (`SPORT_CTL_A.FSR = 0`) not supported
  - `SPORT_CTL_A.DIFS` should be programmed as 0 in transmit mode; while it should be programmed as 1 in receive mode.
- There are few necessary conditions to be satisfied when gated clock mode is enabled-



- Need at least 7 serial clock cycles between enabling the SPORT and first frame sync. If this requirement is not met, the SPORT may drop the first data. (For subsequent data this requirement is not applicable).
- Frame sync should be in the inactive (deasserted) state when the SPORT is enabled. Else, one extra cycle (in addition to the above mentioned) is needed before the frame sync can be applied. If this requirement is not met, the SPORT may drop the first data.
- For edge detected frame sync, the frame sync should transition back to inactive state before the current word transmission/reception is complete (or when the clock is still running). If this requirement is not met, the SPORT does not recognize the next valid frame sync and skips the channel. The SPORT continues to skip the frame syncs until the frame sync transitions back to an inactive state when the clock is active.

## Data Transfers

Serial port data can be transferred to/from internal or external memory in two different methods:

- Core-driven single word transfers
- DMA-driven multiple words transfers, with multiple work units.

DMA transfers can be set up to transfer a configurable number of serial words between the serial port transmit or receive data buffers and internal memory automatically. Core-driven transfers use SPORT interrupts to signal the processor core to perform single word transfers to/from the serial port data buffers.

The following sections provide information on core-driven and DMA-driven data transfers.

## Data Buffers

When programming the serial port data channels (primary and/or secondary) as a transmitter by setting `SPORT_CTL_A.SPTRAN = 1`, only the corresponding transmit data buffers (`SPORT_TXPRI_A` and `SPORT_TXSEC_A`) become active while the receive data buffers (`SPORT_RXPRI_A` and `SPORT_RXSEC_A`) remain inactive. Similarly, when the SPORT data channels are programmed for receive operation (`SPORT_CTL_A.SPTRAN = 0`), then only corresponding receive data buffers (`SPORT_RXPRI_A` and `SPORT_RXSEC_A`) are activated. Do not attempt to read or write inactive data buffers. If the processor operates on the inactive transmit or receive buffers while the SPORT is enabled, unpredictable results may occur.

Each of these buffers is 32-bit wide (corresponds to maximum serial data word length). When using word lengths less than 32-bits for SPORT operation, the data in these buffers is automatically right-justified (the LSB bit of data at the bit 0 location of the buffer). The upper unused bits may be zero-filled or sign-extended depending on `SPORT_CTL_A.DTYPE` field.

## **Transmit Data Buffers (SPORT\_TXPRI\_A and SPORT\_TXSEC\_A)**

When enabled as a transmitter (SPORT\_CTL\_A.SPTRAN = 1), each SPORT half has its own set of transmit data buffers. The primary (0) and secondary (1) data paths of each SPORT half have separate data buffers, referred to as SPORT\_TXPRI\_A and SPORT\_TXSEC\_A respectively.

These transmit data buffers are the 32 bits wide. These buffers must be loaded with the data to be transmitted on the primary and secondary data channels. The data is loaded automatically by the DMA controller or loaded manually by the program running on the processor core.

Together with the output shift register, transmit data buffers act like a two-location FIFO. If data packing is disabled (SPORT\_CTL\_A.PACK = 0), the transmit path can hold as many as three data words. If data packing is enabled (SPORT\_CTL\_A.PACK = 1), it can hold two packed data words at any given time.

When the transmit shift register becomes empty (transfer out all the bits of previous word), data in the transmit data buffer is automatically loaded into it. An interrupt occurs when the output transmit shift register has been loaded, signifying that the transmit data buffer is empty and ready to accept the next word. This interrupt does not occur when serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary data path of a SPORT half is enabled, programs should not write to the inactive secondary transmit data buffer and vice-a-versa. If the core keeps writing to the inactive buffer, the status of that transmit buffer becomes full and this may cause the core to hang indefinitely since data is never transmitted to the output shift register.

## **Receive Data Buffers (SPORT\_RXPRI\_A and SPORT\_RXSEC\_A)**

When enabled as receiver (SPORT\_CTL\_A.SPTRAN = 0), each SPORT half has its own set of receive data buffers. The primary (0) and secondary (1) data paths of each SPORT half have separate data buffers, referred as SPORT\_RXPRI\_A and SPORT\_RXSEC\_A respectively. Together with input shift register, the receive data buffers act like a three-location FIFO, as the receive path has two data registers.

These receive data buffers are the 32 bits wide. These buffers are automatically loaded from the receive shift register when a complete word has been received into it. An interrupt occurs when the receive data buffer is loaded, signifying that new data is available in the receive data buffer and is ready to read. This interrupt does not occur when the serial port is operating in DMA mode or when the corresponding interrupt enable mask bit is set.

If only the primary data path of a SPORT half is enabled, programs should not read from the inactive secondary receive data buffer and vice-a-versa. If the core keeps reading from the inactive buffer, the status of that receive buffer becomes empty and this may cause the core to hang indefinitely since new data is never received via the input shift register.

## **Data Buffer Status**

Serial ports provide status information about data buffers via the SPORT\_CTL\_A.DXSPRI (primary channel data buffer status) and SPORT\_CTL\_A.DXSSEC (secondary channel data buffer status) bits and error status

via `SPORT_CTL_A.DERRPRI` (primary channel error status) and `SPORT_CTL_A.DERRSEC` (secondary channel error status) bits. Depending on the `SPORT_CTL_A.SPTRAN` bit setting, these bits reflect the status of either `SPORT_TXPRI_A` and `SPORT_TXSEC_A` transmit data buffers or `SPORT_RXPRI_A` and `SPORT_RXSEC_A` receive data buffers. These bits indicate whether the buffers are full, partially full or empty.

When attempting to read from an empty receive buffer or to write to a full transmit buffer, the access is delayed until the buffer is ready. This delay is called a core processor hang. To avoid these conditions, always check the buffer status to determine if the access can be made. The status bits in the `SPORT_CTL_A` register are updated during reads and writes from the core processor even when the serial port is disabled.

Two complete 32-bit words can be stored in the receive buffer while a third word is being shifted in. Therefore, almost three complete words can be received without the receive buffer being read before an overflow occurs. After receiving the third word completely, a shift register contents overwrite the second word if the first word has not been read out (by the processor core or the DMA controller). When this happens, the receive overflow status is flagged through the error status bits of the `SPORT_CTL_A` register. The overflow status is generated on the last bit of the third word. The `SPORT_CTL_A.DERRPRI` and `SPORT_CTL_A.DXSSEC` status bits are sticky read-only bits and are cleared by disabling the serial port.

**NOTE:** The status bits in the `SPORT_CTL_A` register are updated during reads and writes from the core processor even when the serial port is disabled.

## Data Buffer Packing

When the SPORT is configured as a receiver with a serial data word length of 16 or less, then received data words may be packed into a 32-bit word. Similarly, if the SPORT is configured as transmitter with a serial data word length of 16 or less, then 32-bit words being transmitted may be unpacked into 16-bit words. This packing/unpacking feature is selected by the `SPORT_CTL_A.PACK` bit.

When `SPORT_CTL_A.PACK = 1`, two successive words received are packed into a single 32-bit word, or each 32-bit word is unpacked and transmitted as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15–0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31–16. This applies to both receive (packing) and transmit (unpacking) operations. In this case, the transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word.

Companding can be used with word packing or unpacking.

**NOTE:** When 16-bit received data is packed into 32-bit words and stored in normal word space in processor internal memory, the 16-bit words can be read or written with short word space addresses.

## Single Word (Core) Transfers

Individual data words may also be transmitted or received by the serial ports, with interrupts occurring as each data word is transmitted or received. When a serial port is enabled and corresponding DMA is disabled, the SPORT interrupts are generated whenever a complete word has been received in the receive data buffer, or whenever the transmit data buffer is not full.

When performing core driven transfers, write to the buffer designated by the `SPORT_CTL_A.SPTRAN` bit setting. For DMA driven transfers, the SPORT logic performs the data transfer from internal memory to/from the appropriate buffer depending on the `SPORT_CTL_A.SPTRAN` bit setting. If the inactive SPORT data buffers are read or written to by the core while the SPORT is being enabled, the core may hang. For example, if a SPORT is programmed to be a transmitter, while at the same time the core reads from the receive buffer of the same SPORT, the core may hang just as it would if it were reading an empty buffer that is currently active and waits for the status to change. This may lock up the core until the SPORT is reset.

To avoid hanging the processor core, check the status of appropriate data buffers when the processor core tries to read a word from a serial port's receive buffer or writes a word to its transmit buffer. The full/empty status can be read using the `SPORT_CTL_A.DXSPRI` and `SPORT_CTL_A.DXSSEC` bits.

## DMA Transfers

Direct memory access (DMA) provides a mechanism for receiving or transmitting an entire block of serial data before an interrupt is generated. When SPORT DMA is not enabled, the SPORT generates an interrupt every time it receives or starts to transmit a data word. The processor's on-chip DMA controller handles the DMA transfer, allowing the processor core to continue running until the entire block of data is transmitted or received. Service routines can then operate on the block of data rather than on single words, significantly reducing overhead.

Therefore, set the direction bit, DMA enable bit and the serial port enable bit before initiating any operations on the SPORT data buffers. Do not try to access data buffers when the associated DMA channel of serial port is enabled. If the processor operates on the inactive transmit or receive buffers while the SPORT is enabled, it can cause unpredictable results.

Each SPORT half has a dedicated DMA channel, which serves both primary and secondary data paths. In transmit mode, the DMA channel alternatively writes to the primary transmit data buffer and the secondary transmit data buffer. Software must interleave the data of primary and secondary channels into the DMA's transmit buffer. Similarly, in receive mode, the DMA channel alternatively reads from the primary receive data buffer and the secondary data buffer and software must de-interleave the data corresponding to the primary and secondary channels from the DMA's receive buffer.

If the SPORT is configured in stereo mode, the same DMA channel drives/receives both the left and right channels of the enabled data paths (primary and/or secondary). Therefore, in transmit mode, software must interleave the left and right channels' data (of the enabled data paths) into the DMA's transmit buffer. Similarly in receive mode, software should de-interleave the left and right channel data (of the enabled paths) from the DMA's receive buffer.

Since both primary and secondary data paths share the single DMA channel, each SPORT half share a common interrupt vector. Optionally the DMA controller can generate an interrupt at the end of the completion of a DMA transfer and at the end of each work unit of DMA.

The SPORT DMA channels are assigned a higher priority than all other DMA channels (for example, the SPI port) because of their relatively low service rate and their inability to hold off incoming data. Having

higher priority causes the SPORT DMA transfers to be performed first when multiple DMA requests occur in the same cycle. The serial port DMA channels are numbered and prioritized as shown in the following table.

**Table 26-14:** SPORT DMA Channel Assignments

| SPORT Channels |              | DMA Channel Number |
|----------------|--------------|--------------------|
| SPORT0         | SPORT half-A | 0                  |
|                | SPORT half-B | 1                  |
| SPORT1         | SPORT half-A | 2                  |
|                | SPORT half-B | 3                  |
| SPORT2         | SPORT half-A | 4                  |
|                | SPORT half-B | 5                  |

Although the DMA transfers are performed with 32-bit words, the SPORTs can handle word sizes from 4 to 32 bits (as defined by `SPORT_CTL_A.SLEN` field). If serial data length is 16 bits or smaller, two data can be packed into 32-bit words for each DMA transfer. This option is selected by setting the `SPORT_CTL_A.PACK` bit. When serial port data packing is enabled (`SPORT_CTL_A.PACK = 1`), the transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word. For more information, see [Data Buffer Status](#).

**NOTE:** The DMA channel of a SPORT can access both internal memory and external memory of the processor without any core overhead.

## Error Detection

When the serial port is configured as a transmitter, the `SPORT_CTL_A.DERRPRI` (primary channel error status) and `SPORT_CTL_A.DERRSEC` (secondary channel error status) bits provide transmit data buffer underflow status for the primary and secondary data paths respectively (it indicates that frame sync signal occurred when the transmit data buffer was empty). The serial port transmits data whenever it detects a framing signal.

- 0 = No frame sync signal occurred when TX data buffer is empty (no underflow).
- 1 = Framing signal occurred when TX buffer was empty (underflow).

Similarly if SPORT configured as a receiver, these `SPORT_CTL_A.DERRPRI` and `SPORT_CTL_A.DERRSEC` bits provide receive overflow status of primary and secondary receive data buffers. In other words it indi-

cates that a channel has received new data when the receive buffer is full so new data overwrites existing data. The serial port receives data whenever it detects a framing signal.

- 0 = No frame sync signal occurred when RX data buffer is full (no overflow).
- 1 = Frame sync signal occurred when RX data buffer was full (overflow).

Besides these status flagging for underflow and overflow errors, each SPORT half contains an error register (SPORT\_ERR\_A) and a dedicated interrupt channel, known as status interrupt. This interrupt can be optionally triggered based on the error status of primary and secondary data lines as reflected in SPORT\_ERR\_A.DERRPSTAT and SPORT\_ERR\_A.DERRSSTAT bits respectively. The SPORT\_ERR\_A.DERRPMSK (primary channel data error interrupt enable) and SPORT\_ERR\_A.DERRSMSK (secondary channel data error interrupt enable) bits can be used to unmask the status interrupt for primary and secondary data errors.

In addition to data underflow and data overflow errors, the status interrupt is also triggered optionally when frame sync error is detected. The SPORT\_ERR\_A.FSERRMSK (frame sync error interrupt enable) bit unmask the status interrupt for this frame sync error. This frame sync error is generated because of premature frame sync, as explained in "Premature Frame sync error detection" section.

Note that a frame sync error is not detected in following cases.

- When there is no active data transmit/receive and the frame sync pulse occurs due to noise in the input signal.
- If there is an underflow or overflow error. SPORT error logic does not run (the bit count is not set and decremented) if there is an underflow error. Therefore, frame sync errors cannot be detected.
- When the frame sync pulse > system clock period.

The channel error status bits, SPORT\_CTL\_A.DERRPRI and SPORT\_CTL\_A.DERRSEC, in the control register are sticky read-only bits, which can be cleared in two ways:

- By disabling the SPORT (for frame sync error) or disabling the corresponding channel by itself (for SPORT\_CTL\_A.DERRPRI, SPORT\_CTL\_A.DERRSEC).
- By writing to R/W the SPORT\_ERR\_A.FSERRSTAT, SPORT\_ERR\_A.DERRPSTAT or SPORT\_ERR\_A.DERRSSTAT status bits.

When sticky bits are cleared, interrupts are also cleared.

## Interrupts

This section describes the various scenarios in which an interrupt is generated. Both the core and DMA are able to generate data interrupts for receive or transmit operations. Moreover, the SPORT modules generate error conditions which have a separate status interrupt.

## Internal Transfer Completion

Each SPORT half has an interrupt associated with it. Both primary and secondary data channels share the same interrupt vector, regardless of whether they are configured as a transmitter or receiver. To determine the source of an interrupt, applications should check the transmit or receive data buffer status (SPORT\_CTL\_A.DXSPRI, SPORT\_CTL\_A.DXSSEC) bits. In core mode, this interrupt signifies that either the transmit data buffer is empty (when SPORT\_CTL\_A.SPTRAN = 1) or new data is available in the receive data buffer (when SPORT\_CTL\_A.SPTRAN = 0). When serial port data packing is enabled (SPORT\_CTL\_A.PACK = 1), the transmit and receive interrupts are generated for 32-bit packed words, not for each 16-bit word.

The same interrupt can be used to indicate the completion of the transfer of a block of serial data when the serial ports are configured for DMA. The count register of DMA must be initialized with a word count that specifies the number of words to transfer. The count register decrements after each DMA transfer on the channel. When the word count reaches zero (or if a work unit has finished), the DMA completion interrupt is generated.

Multiple interrupts can occur if both data channels of serial port transmit or receive data in the same cycle. Any interrupt can be masked in the IMASK register.

## Transfer Finish Interrupt (TFI)

When a serial port in DMA mode is configured as a transmitter (SPORT\_CTL\_A.SPTRAN = 1), the Transmit Finish Interrupt feature can be used to signal the end of the transmission in a particular work unit. This feature can be enabled by setting SPORT\_CTL\_A.TFIENbit. When DMA transfers the last word to the FIFO, it also gives a signal to the SPORT indicating DMA has finished. The SPORT uses this information and then waits until all the data in the FIFO is transmitted out (including the transmit shift register) and generates the Transmit Finish Interrupt. The Interrupt Type field in the DMA Configuration register should be configured for Peripheral Interrupt.

## SPORT Register Descriptions

Serial Port (SPORT) contains the following registers.

**Table 26-15:** SPORT Module Registers

| Name         | Description                                       |
|--------------|---------------------------------------------------|
| SPORT_CTL_A  | Half SPORT 'A' Control Register                   |
| SPORT_DIV_A  | Half SPORT 'A' Divisor Register                   |
| SPORT_MCTL_A | Half SPORT 'A' Multi-channel Control Register     |
| SPORT_CS0_A  | Half SPORT 'A' Multi-channel 0-31 Select Register |

**Table 26-15:** SPORT Module Registers (Continued)

| Name          | Description                                         |
|---------------|-----------------------------------------------------|
| SPORT_CS1_A   | Half SPORT 'A' Multi-channel 32-63 Select Register  |
| SPORT_CS2_A   | Half SPORT 'A' Multi-channel 64-95 Select Register  |
| SPORT_CS3_A   | Half SPORT 'A' Multi-channel 96-127 Select Register |
| SPORT_ERR_A   | Half SPORT 'A' Error Register                       |
| SPORT_MSTAT_A | Half SPORT 'A' Multi-channel Status Register        |
| SPORT_CTL2_A  | Half SPORT 'A' Control 2 Register                   |
| SPORT_TXPRI_A | Half SPORT 'A' Tx Buffer (Primary) Register         |
| SPORT_RXPRI_A | Half SPORT 'A' Rx Buffer (Primary) Register         |
| SPORT_TXSEC_A | Half SPORT 'A' Tx Buffer (Secondary) Register       |
| SPORT_RXSEC_A | Half SPORT 'A' Rx Buffer (Secondary) Register       |
| SPORT_CTL_B   | Half SPORT 'B' Control Register                     |
| SPORT_DIV_B   | Half SPORT 'B' Divisor Register                     |
| SPORT_MCTL_B  | Half SPORT 'B' Multi-channel Control Register       |
| SPORT_CS0_B   | Half SPORT 'B' Multi-channel 0-31 Select Register   |
| SPORT_CS1_B   | Half SPORT 'B' Multi-channel 32-63 Select Register  |
| SPORT_CS2_B   | Half SPORT 'B' Multichannel 64-95 Select Register   |
| SPORT_CS3_B   | Half SPORT 'B' Multichannel 96-127 Select Register  |
| SPORT_ERR_B   | Half SPORT 'B' Error Register                       |
| SPORT_MSTAT_B | Half SPORT 'B' Multi-channel Status Register        |
| SPORT_CTL2_B  | Half SPORT 'B' Control 2 Register                   |
| SPORT_TXPRI_B | Half SPORT 'B' Tx Buffer (Primary) Register         |
| SPORT_RXPRI_B | Half SPORT 'B' Rx Buffer (Primary) Register         |



**Table 26-15:** SPORT Module Registers (Continued)

| Name          | Description                                   |
|---------------|-----------------------------------------------|
| SPORT_TXSEC_B | Half SPORT 'B' Tx Buffer (Secondary) Register |
| SPORT_RXSEC_B | Half SPORT 'B' Rx Buffer (Secondary) Register |

## Half SPORT 'A' Control Register

The SPORT\_CTL\_A contains transmit and receive control bits for SPORT half 'A', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in SPORT\_CTL\_A vary, depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

SPORT\_CTL\_A: Half SPORT 'A' Control Register - R/W

Reset = 0x0000 0000

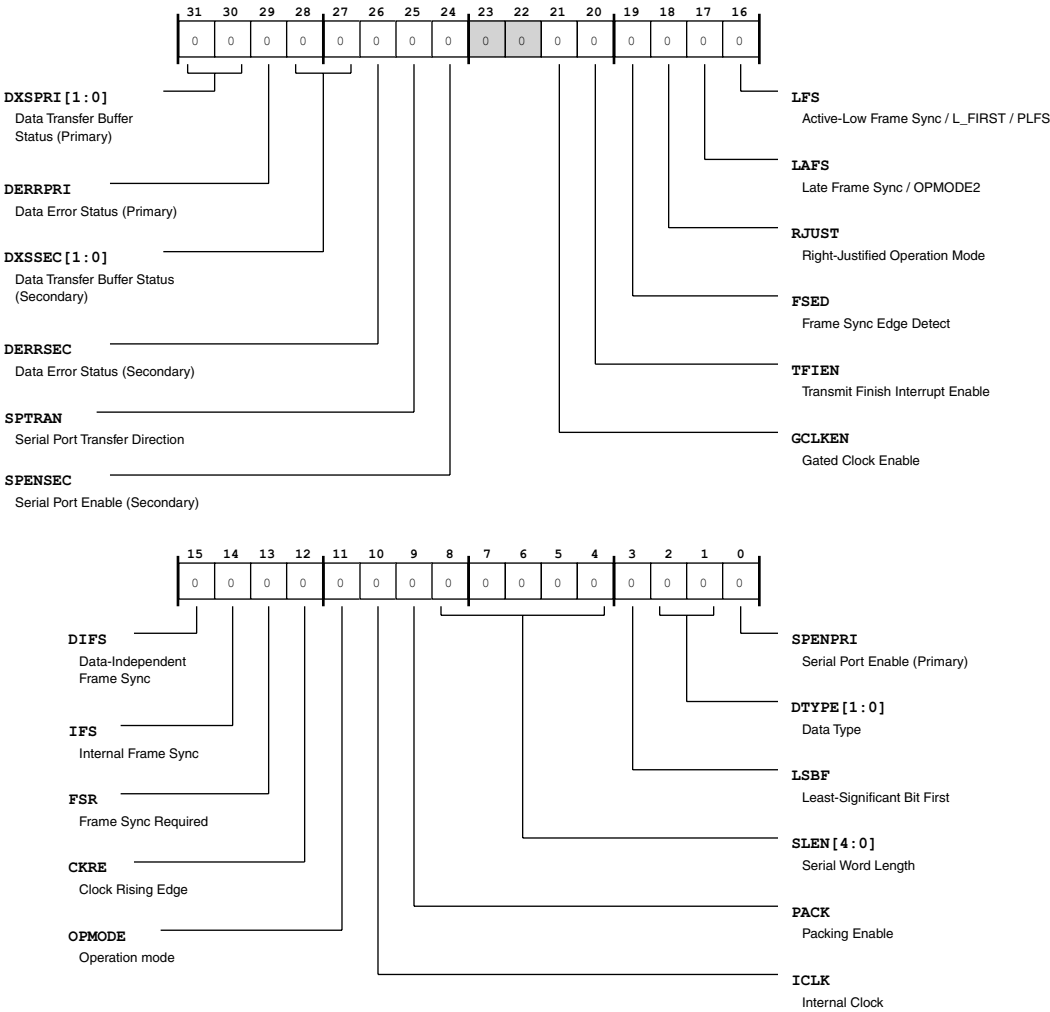


Figure 26-15: SPORT\_CTL\_A Register Diagram

**Table 26-16:** SPORT\_CTL\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:30   | DXSPRI   | Data Transfer Buffer Status (Primary).<br>The SPORT_CTL_A.DXSPRI indicates the status of the half SPORT's primary channel data buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0 Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 2 Partially full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 3 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 29      | DERRPRI  | Data Error Status (Primary).<br>The SPORT_CTL_A.DERRPRI reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction.<br>If the SPORT_CTL_A.FSR bit =1, SPORT_CTL_A.DERRPRI indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXPRI_A data buffer was empty (during transmit) or the SPORT_RXPRI_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRPRI indicates when the channel has received new data while the SPORT_RXPRI_A receive buffer is full. This new data overwrites existing data.<br>If the SPORT_CTL_A.FSR bit =0, SPORT_CTL_A.DERRPRI is set whenever the SPORT is required to transmit while the SPORT_TXPRI_A transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXPRI_A receive buffer is full.<br>The SPORT clears the SPORT_CTL_A.DERRPRI bit if the SPORT_ERR_A.DERRSTAT bit is cleared. |
|         |          | 0 No error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Error (Tx underflow or Rx overflow)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28:27   | DXSSEC   | Data Transfer Buffer Status (Secondary).<br>The SPORT_CTL_A.DXSSEC indicates the status of the half SPORT's secondary channel data buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0 Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2 Partially full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 3 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 26      | DERRSEC  | Data Error Status (Secondary).<br>The SPORT_CTL_A.DERRSEC reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction.<br>If the SPORT_CTL_A.FSR bit =1, SPORT_CTL_A.DERRSEC indicates whether the SPORT_AFS signal (from an internal or external source) occurred while the SPORT_TXSEC_A data buffer was empty (during transmit) or the SPORT_RXSEC_A data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_AFS signal. It is important to note that, as a receiver, the SPORT_CTL_A.DERRSEC indicates when the channel has received new data while the SPORT_RXSEC_A receive buffer is full. This new data overwrites existing data.<br>If the SPORT_CTL_A.FSR bit =0, SPORT_CTL_A.DERRSEC is set whenever the SPORT is required to transmit while the SPORT_TXSEC_A transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXSEC_A receive buffer is full.<br>The SPORT clears the SPORT_CTL_A.DERRSEC bit if the SPORT_ERR_A.DERRSTAT bit is cleared. |
|         |          | 0 No error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Error (Tx underflow or Rx overflow)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | SPTRAN   | <p>Serial Port Transfer Direction.</p> <p>The SPORT_CTL_A.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_ACLK and SPORT_AFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive.</p> <p>When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_ACLK and SPORT_AFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.</p> |
|         |          | 0      Receive                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Transmit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 24      | SPENSEC  | <p>Serial Port Enable (Secondary).</p> <p>The SPORT_CTL_A.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers. The flushing process requires 6 core cycles.</p>                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21      | GCLKEN   | <p>Gated Clock Enable.</p> <p>The SPORT_CTL_A.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_A.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_A.RJUST =1) or multi-channel mode (SPORT_MCTL_A.MCE =1).</p> <p>When SPORT_CTL_A.GCLKEN is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).</p>                                                                                                                                                                  |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | TFIEN    | Transmit Finish Interrupt Enable.<br>The SPORT_CTL_A.TFIEN bit selects when the half SPORT issues its transmission complete interrupt if a DMA complete interrupt is enabled by the DDE_CFG_INT configuration. When enabled (SPORT_CTL_A.TFIEN =1), the DMA complete peripheral interrupt is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_A.TFIEN =0), the DMA interrupt is generated when the DMA counter expires (the last word goes to the transmit buffer).                                                                                                                                             |
|         |          | 0 Last word sent (DMA count done) interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Last bit sent (Tx buffer done) interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 19      | FSED     | Frame Sync Edge Detect.<br>The SPORT_CTL_A.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_A.FSED may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_A.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.                                                                                                                                                                    |
|         |          | 0 Level detect frame sync                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Edge detect frame sync                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 18      | RJUST    | Right-Justified Operation Mode.<br>The SPORT_CTL_A.RJUST bit enables the half SPORT (if SPORT_CTL_A.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_A.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                         |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | LAFS     | Late Frame Sync / OPMODE2.<br>When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0) or in right-justified mode (SPORT_CTL_A.RJUST =1), the SPORT_CTL_A.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_AFS during first data bit) or generates an early frame sync signal (SPORT_AFS during serial clock cycle before first data bit). When the half SPORT is in I2S / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I2S mode. When the half SPORT is in multi-channel mode (SPORT_MCTL_A.MCE =1), the SPORT_CTL_A.LAFS bit is reserved. |                                                                                                                                                                                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Early frame sync<br>(or I2S mode)                                                                                                                                                       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Late frame sync<br>(or left-justified mode)                                                                                                                                             |
| 16      | LFS      | Active-Low Frame Sync / L_FIRST / PLFS.<br>When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.LFS bit selects whether the half SPORT uses active low or active high frame sync. When the half SPORT is in I2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), the SPORT_CTL_A.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.                                                                                                                                                                                                                              |                                                                                                                                                                                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Active high frame sync (DSP standard mode)<br>or rising edge frame sync (multi-channel mode)<br>or right channel first (I2S/packed mode)<br>or left channel first (left-justified mode) |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Active low frame sync (DSP standard mode)<br>or falling edge frame sync (multi-channel mode)<br>or left channel first (I2S/packed mode)<br>or right channel first (left-justified mode) |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 15      | DIFS     | Data-Independent Frame Sync.<br>The SPORT_CTL_A.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by SPORT_DIV_A.FSDIV. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_A.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes. |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Data-dependent frame sync   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Data-independent frame sync |
| 14      | IFS      | Internal Frame Sync.<br>The SPORT_CTL_A.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync.<br>Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.                                                                                                                                                                                                                                                                                                            |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | External frame sync         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Internal frame sync         |
| 13      | FSR      | Frame Sync Required.<br>The SPORT_CTL_A.FSR selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1) or is in multi-channel mode (SPORT_MCTL_A.MCE =1).                                                                                                                                                                                                                                                                            |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No frame sync required      |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Frame sync required         |



**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| 12      | CKRE     | <p>Clock Rising Edge.</p> <p>The SPORT_CTL_A.CKRE selects the rising or falling edge of the SPORT_ACLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_ACLK. Also note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for SPORT_CTL_A.CKRE. This programming drives the internally-generated signals on one edge of SPORT_ACLK and samples the received signals on the opposite edge.</p> |                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Clock falling edge              |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Clock rising edge               |
| 11      | OPMODE   | <p>Operation mode.</p> <p>The SPORT_CTL_A.OPMODE bit selects whether the half SPORT operates in DSP standard / multi-channel mode or operates in I2S / packed / left-justified mode. The mode selection affects the operation of the SPORT_CTL_A.LAFS and SPORT_CTL_A.LFS bits. Also, the SPORT_CTL_A.OPMODE bit enables or disables operation of the SPORT_CTL_A.GCLKEN, SPORT_CTL_A.FSED, SPORT_CTL_A.RJUST, SPORT_CTL_A.DIFS, SPORT_CTL_A.FSR, and SPORT_CTL_A.CKRE bits.</p>                                                                                                          |                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | DSP standard/multi-channel mode |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | I2S/packed/left-justified mode  |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | ICLK     | Internal Clock.<br>When the half SPORT is in DSP standard mode (SPORT_CTL_A.OPMODE =0), the SPORT_CTL_A.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_ACLK clock signal, and the SPORT_ACLK is an output. The SPORT_DIV_A.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_ACLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.                                                                                                                                                                                                                                                           |
|         |          | 0 External clock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Internal clock                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 9       | PACK     | Packing Enable.<br>The SPORT_CTL_A.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15-0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31-16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

**Table 26-16:** SPORT\_CTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                             |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| 8:4     | SLEN     | Serial Word Length.<br>The SPORT_CTL_A.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length.<br>The formula for selecting the word length in bits is:<br>SPORT_CTL_A.SLEN = (serial word length in bits) - 1<br>For DSP standard mode (SPORT_CTL_A.OPMODE =0), use SPORT_CTL_A.SLEN of 3 to 31 bits.<br>For I2S / packed / left-justified mode (SPORT_CTL_A.OPMODE =1), use SPORT_CTL_A.SLEN of 4 to 31 bits. |                                             |
| 3       | LSBF     | Least-Significant Bit First.<br>The SPORT_CTL_A.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.                                                                                                                                                                                                                                                                                                                                   |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | MSB first sent/received (big endian)        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | LSB first sent/received (little endian)     |
| 2:1     | DTYPE    | Data Type.<br>The SPORT_CTL_A.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_A.OPMODE =0).                                                                                                                                                                                                                                                                                                                  |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Right-justify data, zero-fill unused MSBs   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Right-justify data, sign-extend unused MSBs |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | m-law compand data                          |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | A-law compand data                          |
| 0       | SPENPRI  | Serial Port Enable (Primary).<br>The SPORT_CTL_A.SPENPRI bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers. The flushing process requires 6 core cycles.                                                                                                                                                                                                       |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable                                     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable                                      |

Half SPORT 'A' Divisor Register

The SPORT\_DIV\_A contains divisor values that determine frequencies of internally-generated clocks and frame syncs for half SPORT 'A'.

SPORT\_DIV\_A: Half SPORT 'A' Divisor Register - R/W

Reset = 0x0000 0000

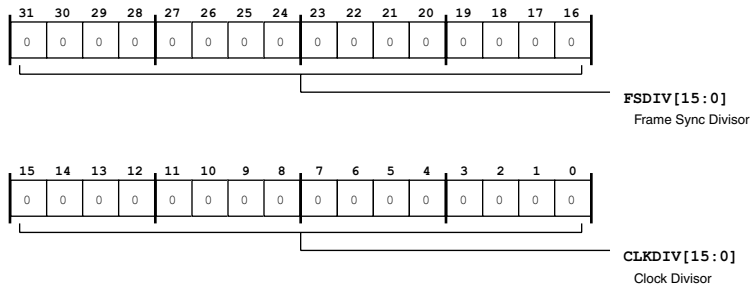


Figure 26-16: SPORT\_DIV\_A Register Diagram

Table 26-17: SPORT\_DIV\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:16   | FSDIV    | <p>Frame Sync Divisor.</p> <p>The SPORT_DIV_A.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_A.FSDIV to the number of cycles between frame sync pulses is:</p> <p><math>SPORT\_DIV\_A.FSDIV = (\text{number of serial clocks between frame syncs}) - 1</math></p> <p>Use the following equation to determine the value of SPORT_DIV_A.FSDIV, given the serial clock frequency and desired frame sync frequency:</p> <p><math>FSDIV = (SCLK / FSCLK) - 1</math></p> <p>Note that the frame sync is continuously active when <math>SPORT\_DIV\_A.FSDIV = 0</math>. The value of SPORT_DIV_A.FSDIV should not be less than the serial word length (SPORT_CTL_A.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.</p> |

**Table 26-17:** SPORT\_DIV\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | CLKDIV   | <p>Clock Divisor.</p> <p>The SPORT_DIV_A.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_ACLK) from the processor system clock (SCLK1). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_A.ICLK=1), legal SPORT_DIV_A.CLKDIV values are 0 to 65535. Given the processor system clock 1 frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_A.CLKDIV:</p> $\text{CLKDIV} = (\text{SCLK1} / \text{SPORT\_ACLK}) - 1$ <p>For the maximum serial clock frequency, see the processor data sheet.</p> |

## Half SPORT 'A' Multi-channel Control Register

The SPORT\_MCTL\_A register controls the half SPORT's multi-channel operations. This register enables multi-channel operation, enables multi-channel data packing, selects the multi-channel frame delay, selects the number of multi-channel slots, and selects the multi-channel window offset size.

### SPORT\_MCTL\_A: Half SPORT 'A' Multi-channel Control Register - R/W

Reset = 0x0000 0000

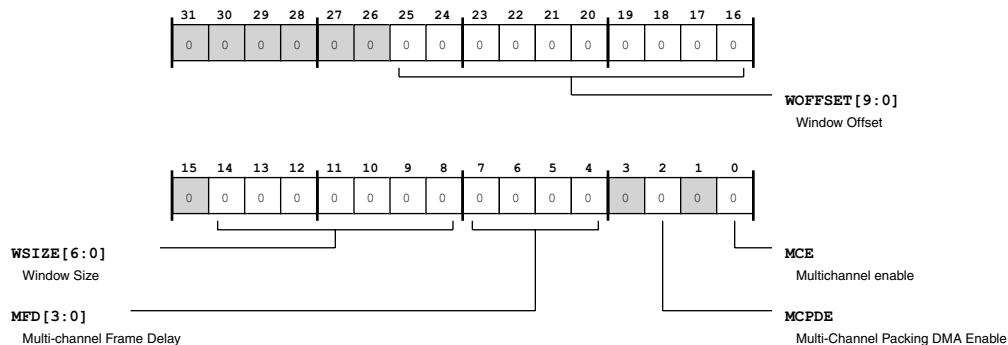


**Figure 26-17:** SPORT\_MCTL\_A Register Diagram

**Table 26-18:** SPORT\_MCTL\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |         |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 25:16   | WOFFSET  | <p>Window Offset.</p> <p>The SPORT_MCTL_A.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multi-channel mode is disabled (SPORT_MCTL_A.MCE =0)and right-justified mode is enabled (SPORT_CTL_A.RJUST =1), the least significant 6 bits of SPORT_MCTL_A.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section.</p> |         |
| 14:8    | WSIZE    | <p>Window Size.</p> <p>The SPORT_MCTL_A.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value:</p> <p>SPORT_MCTL_A.WSIZE = (number of channel slots) -1</p>                                                                                                                                                                                                                                                                                                                                                                                                                             |         |
| 7:4     | MFD      | <p>Multi-channel Frame Delay.</p> <p>The SPORT_MCTL_A.MFD bits select the delay (in serial clock cycles) between the half SPORT's multi-channel frame sync pulse and channel 0. The 4-bit field allows selecting Multichannel Frame delay of 0-15 serial clocks.</p>                                                                                                                                                                                                                                                                                                                                                                                                             |         |
| 2       | MCPDE    | <p>Multi-Channel Packing DMA Enable.</p> <p>The SPORT_MCTL_A.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multi-channel data transfers.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Enable  |

**Table 26-18:** SPORT\_MCTL\_A Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | MCE      | Multichannel enable.<br>The SPORT_MCTL_A.MCE bit enables multi-channel operations for the half SPORT. The Half SPORT is configured in normal multichannel mode if SPORT_CTL_A.OPMODE=0; while it is configured in Packed mode if SPORT_CTL_A.OPMODE=1. When Configuring in these modes, the Multichannel Enable bit (SPORT_MCTL_A.MCE) should be set before enabling SPORT data channel enable bits (SPORT_CTL_A.SPENPRI and/or SPORT_CTL_A.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared (requires 6 CCLK cycles), and the SPORT_CTL_A.DERRPRI and SPORT_CTL_A.DERRSEC bits are cleared. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## Half SPORT 'A' Multi-channel 0-31 Select Register

The SPORT\_CS0\_A register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

**SPORT\_CS0\_A: Half SPORT 'A' Multi-channel 0-31 Select Register - R/W**

Reset = 0x0000 0000

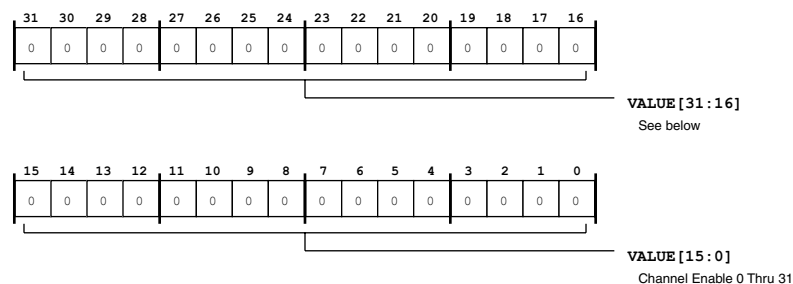


**Figure 26-18:** SPORT\_CS0\_A Register Diagram

**Table 26-19:** SPORT\_CS0\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | VALUE    | Channel Enable 0 Thru 31. |

## Half SPORT 'A' Multi-channel 32-63 Select Register

The SPORT\_CS1\_A register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

**SPORT\_CS1\_A: Half SPORT 'A' Multi-channel 32-63 Select Register - R/W**

Reset = 0x0000 0000

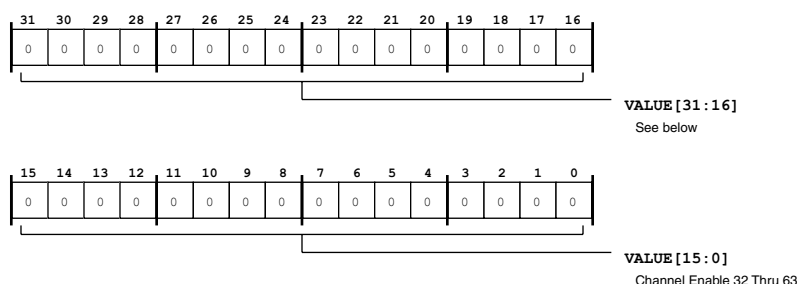


**Figure 26-19:** SPORT\_CS1\_A Register Diagram

**Table 26-20:** SPORT\_CS1\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Channel Enable 32 Thru 63. |

## Half SPORT 'A' Multi-channel 64-95 Select Register

The SPORT\_CS2\_A register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).



**SPORT\_CS2\_A: Half SPORT 'A' Multi-channel 64-95 Select Register - R/W**

Reset = 0x0000 0000

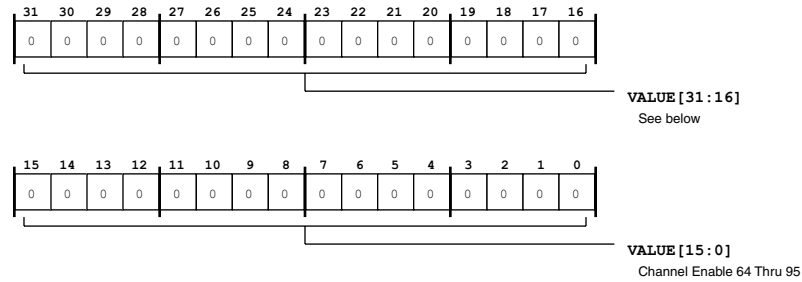


**Figure 26-20:** SPORT\_CS2\_A Register Diagram

**Table 26-21:** SPORT\_CS2\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Channel Enable 64 Thru 95. |

## Half SPORT 'A' Multi-channel 96-127 Select Register

The SPORT\_CS3\_A register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

**SPORT\_CS3\_A: Half SPORT 'A' Multi-channel 96-127 Select Register - R/W**

Reset = 0x0000 0000

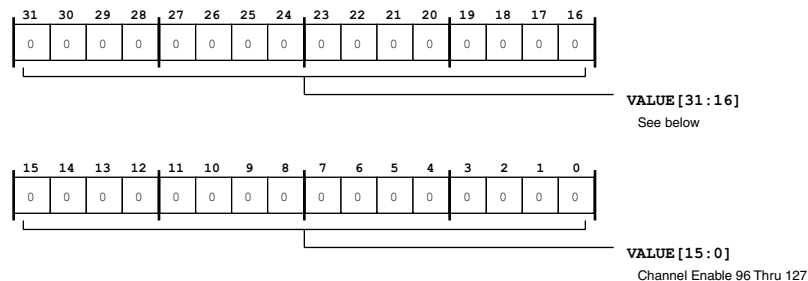


**Figure 26-21:** SPORT\_CS3\_A Register Diagram

Table 26-22: SPORT\_CS3\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 31:0    | VALUE    | Channel Enable 96 Thru 127. |

Half SPORT 'A' Error Register

The SPORT\_ERR\_A contains error status and error interrupt mask bits for SPORT half 'A', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

SPORT\_ERR\_A: Half SPORT 'A' Error Register - R/W

Reset = 0x0000 0000

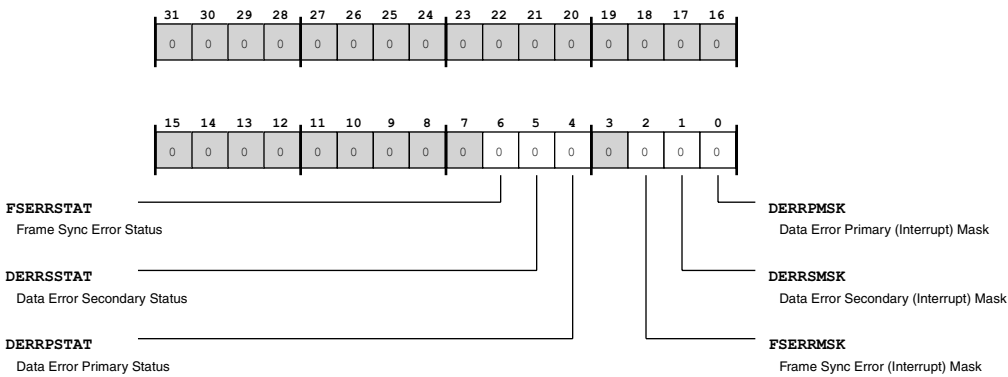


Figure 26-22: SPORT\_ERR\_A Register Diagram

**Table 26-23:** SPORT\_ERR\_A Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| 6       | FSERRSTAT | Frame Sync Error Status.<br>The SPORT_ERR_A.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_A.SLEN = 32). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur during a zero bit count. |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | No error                                       |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Error (non-zero bit count at frame sync)       |
| 5       | DERRSSTAT | Data Error Secondary Status.<br>The SPORT_ERR_A.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), SPORT_ERR_A.DERRSSTAT indicates transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), SPORT_ERR_A.DERRSSTAT indicates receive overflow status. This bit is used to clear the latch of SPORT status interrupt when triggered by secondary Data Error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRSEC status bit.                                                                                                                                                                                            |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | No error                                       |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Error (transmit underflow or receive overflow) |

**Table 26-23:** SPORT\_ERR\_A Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | DERRPSTAT | Data Error Primary Status.<br>The SPORT_ERR_A.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_A.SPTRAN =1), SPORT_ERR_A.DERRPSTAT indicates transmit underflow status. During receive (SPORT_CTL_A.SPTRAN =0), SPORT_ERR_A.DERRPSTAT indicates receive overflow status. This bit is used to clear the latch of SPORT status interrupt when triggered by Primary Data Error. This bit can also be used to clear the read-only SPORT_CTL_A.DERRPRI status bit. |
|         |           | 0 No error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 1 Error (transmit underflow or receive overflow)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 2       | FSERRMSK  | Frame Sync Error (Interrupt) Mask.<br>The SPORT_ERR_A.FSERRMSK unmask (enables) the half SPORT to generate the frame sync error interrupt.                                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 0 Mask (disable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |           | 1 Unmask (enable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 1       | DERRSMSK  | Data Error Secondary (Interrupt) Mask.<br>The SPORT_ERR_A.DERRSMSK unmask (enables) the half SPORT to generate the data error interrupt for the secondary channel.                                                                                                                                                                                                                                                                                                                                                                  |
|         |           | 0 Mask (disable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |           | 1 Unmask (enable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 0       | DERRPMSK  | Data Error Primary (Interrupt) Mask.<br>The SPORT_ERR_A.DERRPMSK unmask (enables) the half SPORT to generate the data error interrupt for the primary channel.                                                                                                                                                                                                                                                                                                                                                                      |
|         |           | 0 Mask (disable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |           | 1 Unmask (enable)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |

## Half SPORT 'A' Multi-channel Status Register

The SPORT\_MSTAT\_A register indicates the current multi-channel being serviced among the half SPORT's active channels in multi-channel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT\_MSTAT\_A register restarts at 0 at each frame sync.

**SPORT\_MSTAT\_A: Half SPORT 'A' Multi-channel Status Register - R/NW**

Reset = 0x0000 0000

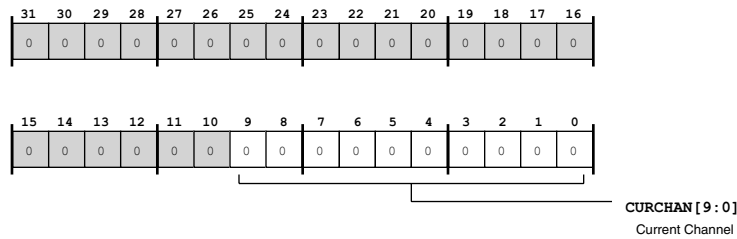


**Figure 26-23:** SPORT\_MSTAT\_A Register Diagram

**Table 26-24:** SPORT\_MSTAT\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 9:0     | CURCHAN  | Current Channel.<br>The SPORT_MSTAT_A.CURCHAN bits indicate the half SPORT's current channel being serviced in multi-channel mode. |

## Half SPORT 'A' Control 2 Register

The SPORT\_CTL2\_A register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

**SPORT\_CTL2\_A: Half SPORT 'A' Control 2 Register - R/W**

Reset = 0x0000 0000

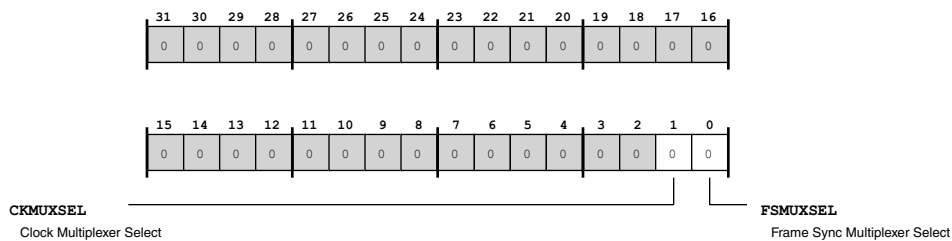


**Figure 26-24:** SPORT\_CTL2\_A Register Diagram

**Table 26-25:** SPORT\_CTL2\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | CKMUXSEL | Clock Multiplexer Select.<br>The SPORT_CTL2_A.CKMUXSEL bit enables multiplexing of the half SPORT' serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if SPORT_CTL2_A.CKMUXSEL is enabled, half SPORT 'A' uses SPORT_BCLK instead of SPORT_ACLK. |
|         |          | 0      Disable serial clock multiplexing                                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable serial clock multiplexing                                                                                                                                                                                                                                                                                                 |
| 0       | FSMUXSEL | Frame Sync Multiplexer Select.<br>The SPORT_CTL2_A.FSMUXSEL bit enables multiplexing of the half SPORT' frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if SPORT_CTL2_A.FSMUXSEL is enabled, half SPORT 'A' uses SPORT_BFS instead of SPORT_AFS.    |
|         |          | 0      Disable frame sync multiplexing                                                                                                                                                                                                                                                                                                  |
|         |          | 1      Enable frame sync multiplexing                                                                                                                                                                                                                                                                                                   |

## Half SPORT 'A' Tx Buffer (Primary) Register

The SPORT\_TXPRI\_A register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core may load the data into the buffer (word-by-word process) or the DMA controller may automatically load the data into the buffer (DMA process).

The SPORT\_TXPRI\_A register acts as a three location buffer if SPORT data packing is disabled (SPORT\_CTL\_A.PACK=0); while it acts as two location buffer when packing is enabled (SPORT\_CTL\_A.PACK=1). So depending on PACK bit setting, two 32-bit words or three 32-bit words may be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT\_TXPRI\_A register contents are automatically loaded into the output shifter. The half SPORT may issue an interrupt (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt does not occur when the half SPORT is executing a DMA-based transfer, and the interrupt may be masked depending how its trigger output is configured in the TRU\_SSRn registers or TRU\_MTR register.

**SPORT\_TXPRI\_A: Half SPORT 'A' Tx Buffer (Primary) Register - R/W**

Reset = 0x0000 0000

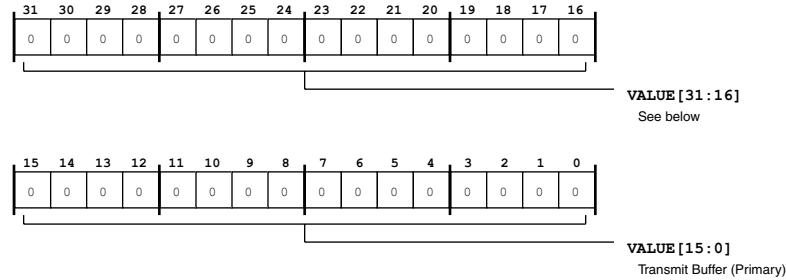


**Figure 26-25:** SPORT\_TXPRI\_A Register Diagram

**Table 26-26:** SPORT\_TXPRI\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Transmit Buffer (Primary).<br>The SPORT_TXPRI_A.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions. |

## Half SPORT 'A' Rx Buffer (Primary) Register

The SPORT\_RXPRI\_A register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. When configured as a receiver, the half SPORT automatically loads the SPORT\_RXPRI\_A register from the receive shifter after a complete word has been received. The half SPORT then loads the data to internal memory through the DMA controller (DMA process) or through the SPORT process of a program running on the processor core (word-by-word process). With a data buffer and input shift register, SPORT\_RXPRI\_A register acts as a two-location buffer. Therefore SPORT can keep Two 32-bit received words at max in any one time (independent of SPORT\_CTL\_A.PACK bit setting).

SPORT\_RXPRI\_A: Half SPORT 'A' Rx Buffer (Primary) Register - R/W

Reset = 0x0000 0000

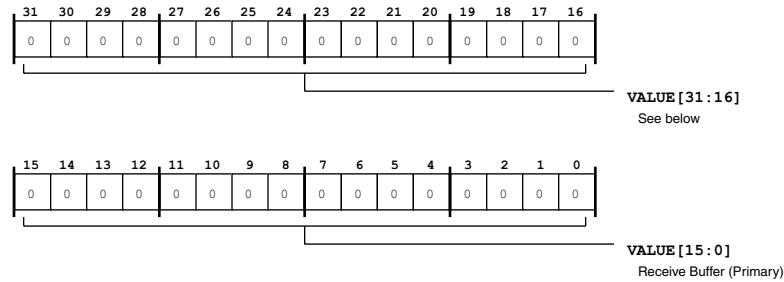


Figure 26-26: SPORT\_RXPRI\_A Register Diagram

Table 26-27: SPORT\_RXPRI\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Receive Buffer (Primary).<br>The SPORT_RXPRI_A.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions. |

Half SPORT 'A' Tx Buffer (Secondary) Register

The SPORT\_TXSEC\_A register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core may load the data into the buffer (word-by-word process) or the DMA controller may automatically load the data into the buffer (DMA process).

The SPORT\_TXSEC\_A register acts as a three location buffer if SPORT data packing is disabled (SPORT\_CTL\_A.PACK=0); while it acts as two location buffer when packing is enabled (SPORT\_CTL\_A.PACK=1). So depending on PACK bit setting, two 32-bit words or three 32-bit words may be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT\_TXSEC\_A register contents are automatically loaded into the output shifter. The half SPORT may issue an interrupt (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt does not occur when the half SPORT is executing a DMA-based transfer, and the interrupt may be masked depending how its trigger output is configured in the TRU\_SSRn registers or TRU\_MTR register.



**SPORT\_TXSEC\_A: Half SPORT 'A' Tx Buffer (Secondary) Register - R/W**

Reset = 0x0000 0000

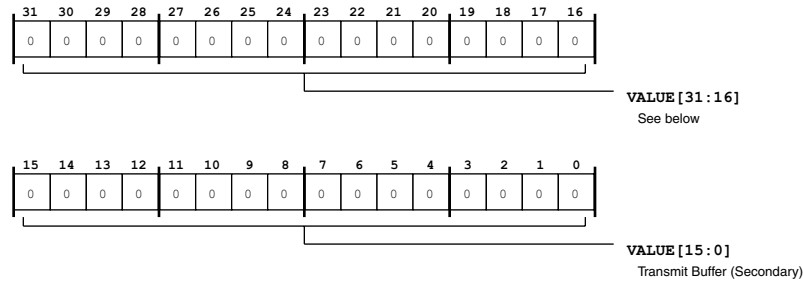


**Figure 26-27:** SPORT\_TXSEC\_A Register Diagram

**Table 26-28:** SPORT\_TXSEC\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Transmit Buffer (Secondary).<br>The SPORT_TXSEC_A.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions. |

## Half SPORT 'A' Rx Buffer (Secondary) Register

The SPORT\_RXSEC\_A register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. When configured as a receiver, the half SPORT automatically loads the SPORT\_RXSEC\_A register from the receive shifter after a complete word has been received. The half SPORT then loads the data to internal memory through the DMA controller (DMA process) or through the SPORT process of a program running on the processor core (word-by-word process). With a data buffer and input shift register, SPORT\_RXSEC\_A register acts as a two-location buffer. Therefore SPORT can keep Two 32-bit received words at max in any one time (independent of SPORT\_CTL\_A.PACK bit setting).

SPORT\_RXSEC\_A: Half SPORT 'A' Rx Buffer (Secondary) Register - R/W

Reset = 0x0000 0000

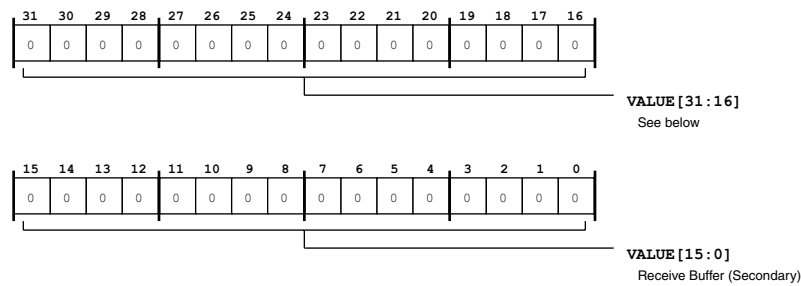


Figure 26-28: SPORT\_RXSEC\_A Register Diagram

Table 26-29: SPORT\_RXSEC\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Receive Buffer (Secondary).<br>The SPORT_RXSEC_A.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_A.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_A and SPORT_MCTL_A register descriptions. |

Half SPORT 'B' Control Register

The SPORT\_CTL\_B contains transmit and receive control bits for SPORT half 'B', including serial port mode selection for the half SPORT's primary and secondary channels. The function of some bits in SPORT\_CTL\_B vary, depending on the SPORT's operating mode. For more information, see the SPORT operating modes description. If reading reserved bits, the read value is the last written value to these bits or is the reset value of these bits.

**SPORT\_CTL\_B: Half SPORT 'B' Control Register - R/W**

Reset = 0x0000 0000

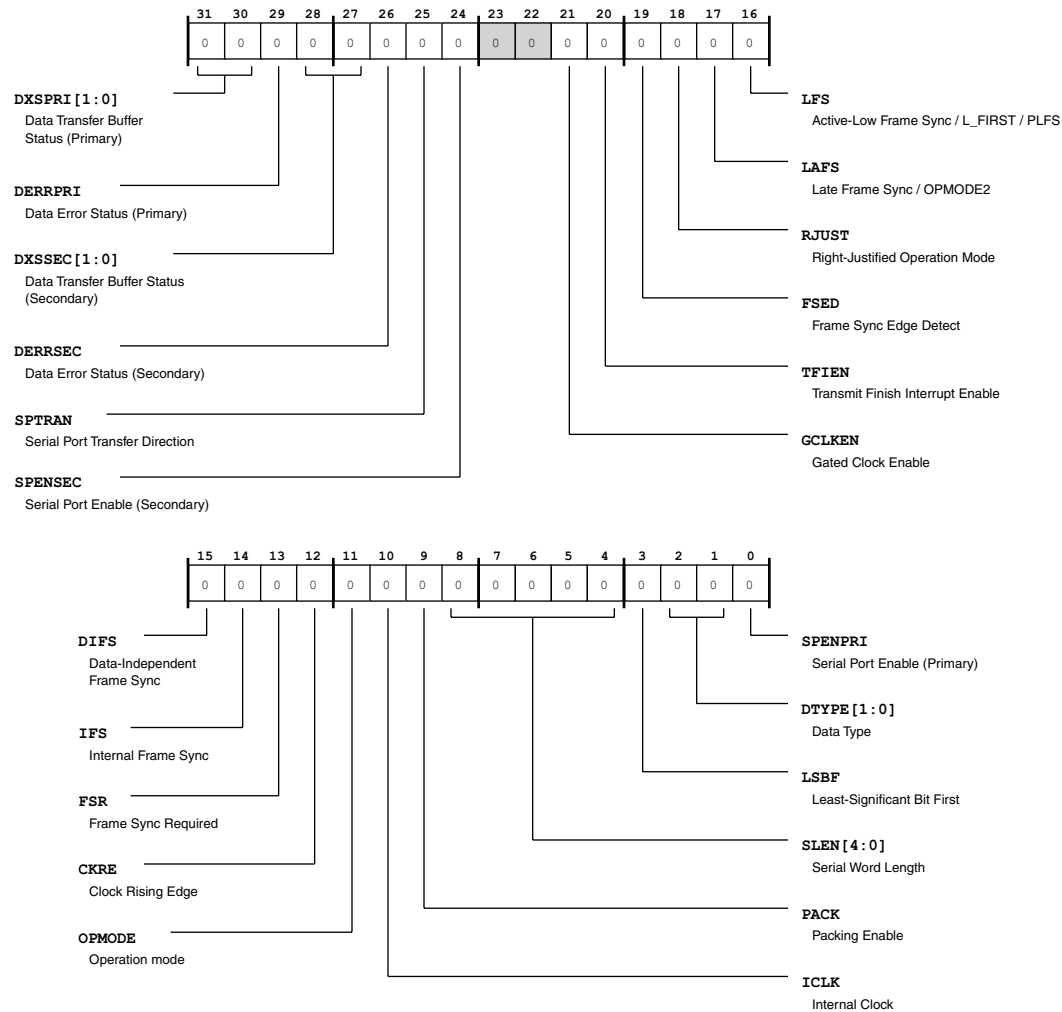


**Figure 26-29: SPORT\_CTL\_B Register Diagram**

**Table 26-30:** SPORT\_CTL\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:30   | DXSPRI   | Data Transfer Buffer Status (Primary).<br>The SPORT_CTL_B.DXSPRI indicates the status of the half SPORT's primary channel data buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0 Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 2 Partially full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 3 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 29      | DERRPRI  | Data Error Status (Primary).<br>The SPORT_CTL_B.DERRPRI reports the half SPORT's primary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction.<br>If the SPORT_CTL_B.FSR bit =1, SPORT_CTL_B.DERRPRI indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXPRI_B data buffer was empty (during transmit) or the SPORT_RXPRI_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRPRI indicates when the channel has received new data while the SPORT_RXPRI_B receive buffer is full. This new data overwrites existing data.<br>If the SPORT_CTL_B.FSR bit =0, SPORT_CTL_B.DERRPRI is set whenever the SPORT is required to transmit while the SPORT_TXPRI_B transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXPRI_B receive buffer is full.<br>The SPORT clears the SPORT_CTL_B.DERRPRI bit if the SPORT_ERR_B.DERRSTAT bit is cleared. |
|         |          | 0 No error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Error (Tx underflow or Rx overflow)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28:27   | DXSSEC   | Data Transfer Buffer Status (Secondary).<br>The SPORT_CTL_B.DXSSEC indicates the status of the half SPORT's secondary channel data buffer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0 Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 2 Partially full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 3 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 26      | DERRSEC  | Data Error Status (Secondary).<br>The SPORT_CTL_B.DERRSEC reports the half SPORT's secondary channel transmit underflow status or receive overflow status, depending on the SPORT transfer direction.<br>If the SPORT_CTL_B.FSR bit =1, SPORT_CTL_B.DERRSEC indicates whether the SPORT_BFS signal (from an internal or external source) occurred while the SPORT_TXSEC_B data buffer was empty (during transmit) or the SPORT_RXSEC_B data buffer was full (during receive). The SPORT transmits or receives data whenever it detects the SPORT_BFS signal. It is important to note that, as a receiver, the SPORT_CTL_B.DERRSEC indicates when the channel has received new data while the SPORT_RXSEC_B receive buffer is full. This new data overwrites existing data.<br>If the SPORT_CTL_B.FSR bit =0, SPORT_CTL_B.DERRSEC is set whenever the SPORT is required to transmit while the SPORT_TXSEC_B transmit buffer is empty and is set whenever the SPORT is required to receive while the SPORT_RXSEC_B receive buffer is full.<br>The SPORT clears the SPORT_CTL_B.DERRSEC bit if the SPORT_ERR_B.DERRSTAT bit is cleared. |
|         |          | 0 No error                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Error (Tx underflow or Rx overflow)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 25      | SPTRAN   | <p>Serial Port Transfer Direction.</p> <p>The SPORT_CTL_B.SPTRAN bit selects the transfer direction (receive or transmit) for the half SPORT's primary and secondary channels. When the direction is receive, the half SPORT activates the receive buffers, and the SPORT_BCLK and SPORT_BFS pins control the receive buffers. The transmit buffers are inactive when the half SPORT's transfer direction is receive.</p> <p>When the direction is transmit, the half SPORT activates the transmit buffers, and the SPORT_BCLK and SPORT_BFS pins control the transmit shift registers. The receive buffers are inactive when the half SPORT's transfer direction is transmit.</p> |
|         |          | 0      Receive                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Transmit                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 24      | SPENSEC  | <p>Serial Port Enable (Secondary).</p> <p>The SPORT_CTL_B.SPENSEC bit enables the half SPORT's secondary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers. The flushing process requires 6 core cycles.</p>                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21      | GCLKEN   | <p>Gated Clock Enable.</p> <p>The SPORT_CTL_B.GCLKEN bit enables gated clock operation for the half SPORT when in DSP serial mode or left-justified stereo modes (SPORT_CTL_B.OPMODE = 0 or 1). This bit is ignored when the half SPORT is in right-justified mode (SPORT_CTL_B.RJUST =1) or multi-channel mode (SPORT_MCTL_B.MCE =1).</p> <p>When SPORT_CTL_B.GCLKEN is enabled, the SPORT clock is active when the SPORT is transferring data or when the frame sync changes (transitions to active state).</p>                                                                                                                                                                  |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 20      | TFIEN    | Transmit Finish Interrupt Enable.<br>The SPORT_CTL_B.TFIEN bit selects when the half SPORT issues its transmission complete interrupt if a DMA complete interrupt is enabled by the DDE_CFG_INT configuration. When enabled (SPORT_CTL_B.TFIEN =1), the DMA complete peripheral interrupt is generated when the last bit of last word in the DMA is shifted out. When disabled (SPORT_CTL_B.TFIEN =0), the DMA interrupt is generated when the DMA counter expires (the last word goes to the transmit buffer).                                                                                                                                             |
|         |          | 0 Last word sent (DMA count done) interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Last bit sent (Tx buffer done) interrupt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 19      | FSED     | Frame Sync Edge Detect.<br>The SPORT_CTL_B.FSED bit enables the half SPORT to start transmitting or receiving after detecting an active edge of an external frame sync. The SPORT_CTL_B.FSED may be enabled even during an active frame sync, and the half SPORT starts the transfer on the next valid rising or falling edge of external frame sync. If disabled (SPORT_CTL_B.FSED =0), the half SPORT operates in the standard level-sensitive detection mode for external frame sync.                                                                                                                                                                    |
|         |          | 0 Level detect frame sync                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Edge detect frame sync                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 18      | RJUST    | Right-Justified Operation Mode.<br>The SPORT_CTL_B.RJUST bit enables the half SPORT (if SPORT_CTL_B.OPMODE =1) to transfer data in right-justified operation mode. In this mode, the half SPORT aligns data to the end of the frame sync, rather than the start of the frame sync. When using right-justified mode, systems should program an appropriate delay count to introduce a clock delay before the half SPORT state machine starts to capture data. This value is set in the DCNT field (right-justified mode usage of the SPORT_MCTL_B.WOFFSET field). For information about appropriate delay selections, see the SPORT operating modes section. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                         |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 17      | LAFS     | Late Frame Sync / OPMODE2.<br>When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0) or in right-justified mode (SPORT_CTL_B.RJUST =1), the SPORT_CTL_B.LAFS bit selects whether the half SPORT generates a late frame sync (SPORT_BFS during first data bit) or generates an early frame sync signal (SPORT_BFS during serial clock cycle before first data bit).<br>When the half SPORT is in I2S / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LAFS bit acts as OPMODE2, selecting whether the half SPORT is in left-justified mode or I2S mode.<br>When the half SPORT is in multi-channel mode (SPORT_MCTL_B.MCE =1), the SPORT_CTL_B.LAFS bit is reserved. |                                                                                                                                                                                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Early frame sync<br>(or I2S mode)                                                                                                                                                       |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Late frame sync<br>(or left-justified mode)                                                                                                                                             |
| 16      | LFS      | Active-Low Frame Sync / L_FIRST / PLFS.<br>When the half SPORT is in DSP standard mode and multichannel mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.LFS bit selects whether the half SPORT uses active low or active high frame sync.<br>When the half SPORT is in I2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), the SPORT_CTL_B.LFS bit acts as L_FIRST, selecting whether the half SPORT transfers data first for the left or right channel.                                                                                                                                                                                                                                 |                                                                                                                                                                                         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Active high frame sync (DSP standard mode)<br>or rising edge frame sync (multi-channel mode)<br>or right channel first (I2S/packed mode)<br>or left channel first (left-justified mode) |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Active low frame sync (DSP standard mode)<br>or falling edge frame sync (multi-channel mode)<br>or left channel first (I2S/packed mode)<br>or right channel first (left-justified mode) |



**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| 15      | DIFS     | Data-Independent Frame Sync.<br>The SPORT_CTL_B.DIFS bit selects whether the half SPORT uses a data-independent or data-dependent frame sync. When using a data-independent frame sync, the half SPORT generates the sync at the interval selected by SPORT_DIV_B.FSDIV. When using a data-dependent frame sync, the half SPORT generates the sync on the selected interval when the transmit buffer is not empty or when the receive buffer is not full. Note that the SPORT_CTL_B.DIFS bit is automatically set when the half SPORT is in packed or multichannel modes. |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Data-dependent frame sync   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Data-independent frame sync |
| 14      | IFS      | Internal Frame Sync.<br>The SPORT_CTL_B.IFS bit selects whether the half SPORT uses an internal frame sync or uses an external frame sync.<br>Note that the externally-generated frame sync does not need to be synchronous with the processor's system clock.                                                                                                                                                                                                                                                                                                            |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | External frame sync         |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Internal frame sync         |
| 13      | FSR      | Frame Sync Required.<br>The SPORT_CTL_B.FSR selects whether or not the half SPORT requires frame sync for data transfer. This bit is automatically set when the half SPORT is in I2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1) or is in multi-channel mode (SPORT_MCTL_B.MCE =1).                                                                                                                                                                                                                                                                            |                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | No frame sync required      |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Frame sync required         |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |                                 |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| 12      | CKRE     | Clock Rising Edge.<br>The SPORT_CTL_B.CKRE selects the rising or falling edge of the SPORT_BCLK clock for the half SPORT to sample receive data and frame sync. Note that the half SPORT changes the state of transmit data and frame sync signals on the non-selected edge of the SPORT_BCLK. Also note that the transmit and receive related SPORT halves (A and B) should be programmed with the same value for SPORT_CTL_B.CKRE. This programming drives the internally-generated signals on one edge of SPORT_BCLK and samples the received signals on the opposite edge. |                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Clock falling edge              |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Clock rising edge               |
| 11      | OPMODE   | Operation mode.<br>The SPORT_CTL_B.OPMODE bit selects whether the half SPORT operates in DSP standard / multi-channel mode or operates in I2S / packed / left-justified mode. The mode selection affects the operation of the SPORT_CTL_B.LAFS and SPORT_CTL_B.LFS bits. Also, the SPORT_CTL_B.OPMODE bit enables or disables operation of the SPORT_CTL_B.GCLKEN, SPORT_CTL_B.FSED, SPORT_CTL_B.RJUST, SPORT_CTL_B.DIFS, SPORT_CTL_B.FSR, and SPORT_CTL_B.CKRE bits.                                                                                                          |                                 |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | DSP standard/multi-channel mode |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | I2S/packed/left-justified mode  |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 10      | ICLK     | Internal Clock.<br>When the half SPORT is in DSP standard mode (SPORT_CTL_B.OPMODE =0), the SPORT_CTL_B.ICLK bit selects whether the half SPORT uses an internal or external clock. For internal clock enabled, the half SPORT generates the SPORT_BCLK clock signal, and the SPORT_BCLK is an output. The SPORT_DIV_B.CLKDIV serial clock divisor value determines the clock frequency. For internal clock disabled, the SPORT_BCLK clock signal is an input, and the serial clock divisor is ignored. Note that the externally-generated serial clock does not need to be synchronous with the processor's system clock.                                                                                                                                                                                                                                                           |                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | External clock |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Internal clock |
| 9       | PACK     | Packing Enable.<br>The SPORT_CTL_B.PACK bit enables the half SPORT to perform 16- to 32-bit packing on received data and to perform 32- to 16-bit unpacking on transmitted data. The receive packing operation packs two successive received words into a single 32-bit word. The transmit unpacking operation unpacks each 32-bit word and transmits it as two 16-bit words. The first 16-bit (or smaller) word is right-justified in bits 15-0 of the packed word, and the second 16-bit (or smaller) word is right-justified in bits 31-16. This format applies to both receive (packing) and transmit (unpacking) operations. Companding may be used with word packing or unpacking. The half SPORT generates data transfer related interrupts when packing is enabled. The transmit and receive interrupts are generated for the 32-bit packed words, not for each 16-bit word. |                |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Disable        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Enable         |

**Table 26-30:** SPORT\_CTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                             |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| 8:4     | SLEN     | Serial Word Length.<br>The SPORT_CTL_B.SLEN bits selects word length in bits for the half SPORT's data transfers. Word may be from 4- to 32-bits in length.<br>The formula for selecting the word length in bits is:<br>SPORT_CTL_B.SLEN = (serial word length in bits) - 1<br>For DSP standard mode (SPORT_CTL_B.OPMODE =0), use SPORT_CTL_B.SLEN of 3 to 31 bits.<br>For I2S / packed / left-justified mode (SPORT_CTL_B.OPMODE =1), use SPORT_CTL_B.SLEN of 4 to 31 bits. |                                             |
| 3       | LSBF     | Least-Significant Bit First.<br>The SPORT_CTL_B.LSBF bit selects whether the half SPORT transmits or receives data LSB first or MSB first.                                                                                                                                                                                                                                                                                                                                   |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | MSB first sent/received (big endian)        |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | LSB first sent/received (little endian)     |
| 2:1     | DTYPE    | Data Type.<br>The SPORT_CTL_B.DTYPE bits selects the data type formatting for the half SPORT's data transfers in DSP standard mode (SPORT_CTL_B.OPMODE =0).                                                                                                                                                                                                                                                                                                                  |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Right-justify data, zero-fill unused MSBs   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Right-justify data, sign-extend unused MSBs |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | m-law compand data                          |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | A-law compand data                          |
| 0       | SPENPRI  | Serial Port Enable (Primary).<br>The SPORT_CTL_B.SPENPRI bit enables the half SPORT's primary channel. When this bit is cleared (changes from =1 to =0), the half SPORT automatically flushes the channel's data buffers. The flushing process requires 6 core cycles.                                                                                                                                                                                                       |                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Disable                                     |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Enable                                      |

## Half SPORT 'B' Divisor Register

The SPORT\_DIV\_B contains divisor values that determine frequencies of internally-generated clocks and frame syncs for SPORT half 'B'.

**SPORT\_DIV\_B: Half SPORT 'B' Divisor Register - R/W**

Reset = 0x0000 0000

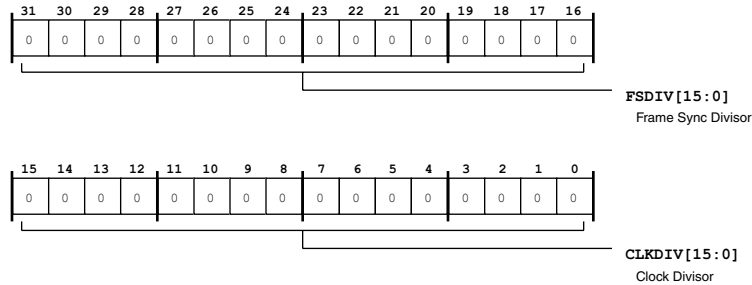


**Figure 26-30:** SPORT\_DIV\_B Register Diagram

**Table 26-31:** SPORT\_DIV\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:16   | FSDIV    | <p>Frame Sync Divisor.</p> <p>The SPORT_DIV_B.FSDIV bits select the number of transmit or receive clock cycles that the half SPORT counts before generating a frame sync pulse. The half SPORT counts serial clock cycles whether these are from an internally- or an externally-generated serial clock. The formula relating SPORT_DIV_B.FSDIV to the number of cycles between frame sync pulses is:</p> $\text{SPORT\_DIV\_B.FSDIV} = (\text{number of serial clocks between frame syncs}) - 1$ <p>Use the following equation to determine the value of SPORT_DIV_B.FSDIV, given the serial clock frequency and desired frame sync frequency:</p> $\text{FSDIV} = (\text{SCLK} / \text{FSCLK}) - 1$ <p>Note that the frame sync is continuously active when SPORT_DIV_B.FSDIV = 0. The value of SPORT_DIV_B.FSDIV should not be less than the serial word length (SPORT_CTL_B.SLEN), as this may cause an external device to abort the current operation or cause other unpredictable results.</p> |

Table 26-31: SPORT\_DIV\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | CLKDIV   | <p>Clock Divisor.</p> <p>The SPORT_DIV_B.CLKDIV bits select the divisor that the half SPORT uses to calculate the serial clock (SPORT_BCLK) from the processor system clock (SCLK1). The divisor is a 16-bit value, allowing a wide range of serial clock rates. When configured for internal clock (SPORT_CTL_B.ICLK =1), legal SPORT_DIV_B.CLKDIV values are 0 to 65535. Given the processor system clock 1 frequency and desired serial clock frequency, use the following formula to calculate the value of SPORT_DIV_B.CLKDIV:</p> $\text{CLKDIV} = (\text{SCLK1} / \text{SPORT\_BCLK}) - 1$ <p>For the maximum serial clock frequency, see the processor data sheet.</p> |

Half SPORT 'B' Multi-channel Control Register

The SPORT\_MCTL\_B register controls the half SPORT's multi-channel operations. This register enables multi-channel operation, enables multi-channel data packing, selects the multi-channel frame delay, selects the number of multi-channel slots, and selects the multi-channel window offset size.

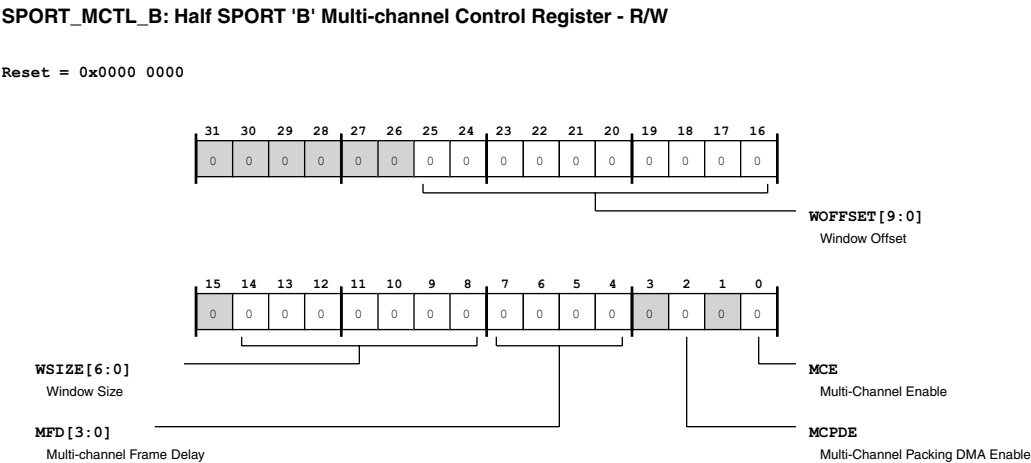


Figure 26-31: SPORT\_MCTL\_B Register Diagram

**Table 26-32:** SPORT\_MCTL\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| 25:16   | WOFFSET  | Window Offset.<br>The SPORT_MCTL_B.WOFFSET bits select the start location for the half SPORT's active window of channels within the 1024-channel range. A value of 0 specifies no offset and 896 is the largest value that permits using all 128 channels. When multi-channel mode is disabled (SPORT_MCTL_B.MCE =0)and right-justified mode is enabled (SPORT_CTL_B.RJUST =1), the least significant 6 bits of SPORT_MCTL_B.WOFFSET serve as the delay count (DCNT) field. These bits introduce a clock delay before the half SPORT state machine starts to capture data. For information about appropriate delay selections, see the SPORT operating modes section. |         |
| 14:8    | WSIZE    | Window Size.<br>The SPORT_MCTL_B.WSIZE bits select the window size for the half SPORT's active window of channels. Use the following formula to calculate the window size value:<br>SPORT_MCTL_B.WSIZE = (number of channel slots) -1                                                                                                                                                                                                                                                                                                                                                                                                                                 |         |
| 7:4     | MFD      | Multi-channel Frame Delay.<br>The SPORT_MCTL_B.MFD bits select the delay (in serial clock cycles) between the half SPORT's multi-channel frame sync pulse and channel 0. The 4-bit field allows selecting Multichannel Frame delay of 0-15 serial clocks.                                                                                                                                                                                                                                                                                                                                                                                                             |         |
| 2       | MCPDE    | Multi-Channel Packing DMA Enable.<br>The SPORT_MCTL_B.MCPDE bit enables DMA data packing for transmit and enables DMA data unpacking for the half SPORT's multi-channel data transfers.                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |         |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Disable |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Enable  |

Table 26-32: SPORT\_MCTL\_B Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0       | MCE      | Multi-Channel Enable.<br>The SPORT_MCTL_B.MCE bit enables multi-channel operations for the half SPORT. The Half SPORT is configured in normal multichannel mode if SPORT_CTL_B.OPMODE=0; while it is configured in Packed mode if SPORT_CTL_B.OPMODE=1. When Configuring in these modes, the Multichannel Enable bit (SPORT_MCTL_B.MCE) should be set before enabling SPORT data channel enable bits (SPORT_CTL_B.SPENPRI and/or SPORT_CTL_B.SPENSEC). When these channel bits transition from 1 to 0, note that the half SPORT's data transfer buffers are cleared (requires 6 CCLK cycles), and the SPORT_CTL_B.DERRPRI and SPORT_CTL_B.DERRSEC bits are cleared. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

Half SPORT 'B' Multi-channel 0-31 Select Register

The SPORT\_CS0\_B register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

SPORT\_CS0\_B: Half SPORT 'B' Multi-channel 0-31 Select Register - R/W

Reset = 0x0000 0000

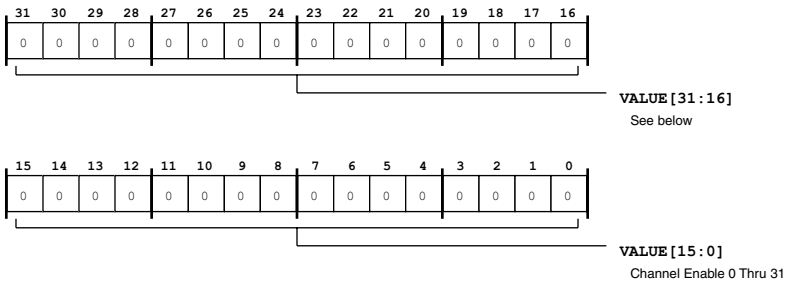


Figure 26-32: SPORT\_CS0\_B Register Diagram



**Table 26-33:** SPORT\_CS0\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | VALUE    | Channel Enable 0 Thru 31. |

## Half SPORT 'B' Multi-channel 32-63 Select Register

The SPORT\_CS1\_B register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

**SPORT\_CS1\_B: Half SPORT 'B' Multi-channel 32-63 Select Register - R/W**

Reset = 0x0000 0000

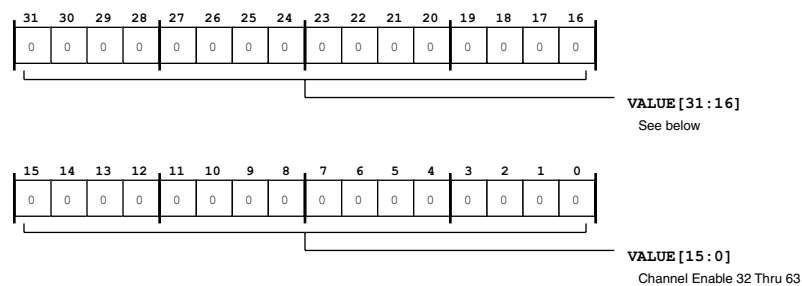


**Figure 26-33:** SPORT\_CS1\_B Register Diagram

**Table 26-34:** SPORT\_CS1\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Channel Enable 32 Thru 63. |

## Half SPORT 'B' Multichannel 64-95 Select Register

The SPORT\_CS2\_B register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

SPORT\_CS2\_B: Half SPORT 'B' Multichannel 64-95 Select Register - R/W

Reset = 0x0000 0000

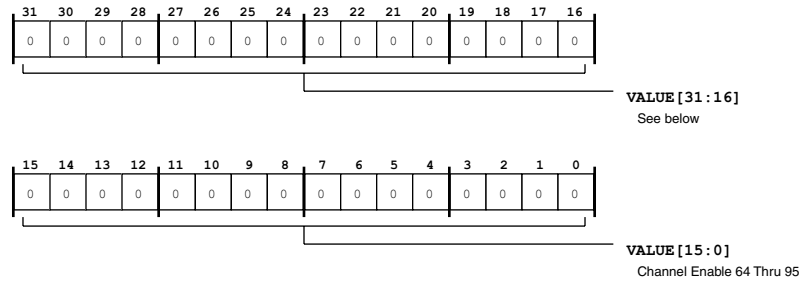


Figure 26-34: SPORT\_CS2\_B Register Diagram

Table 26-35: SPORT\_CS2\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 31:0    | VALUE    | Channel Enable 64 Thru 95. |

Half SPORT 'B' Multichannel 96-127 Select Register

The SPORT\_CS3\_B register's bits (when set, =1) each corresponds to an active channel for the half SPORT in multi-channel mode. When the register activates a channel (corresponding bit =1), the half SPORT transmits or receives the word in that channel's position of the data stream. When the register de-activates a channel (corresponding bit =0), the half SPORT either three-states its data transmit pin (during the channel's transmit time slot) or ignores incoming data (during the channel's receive time slot).

SPORT\_CS3\_B: Half SPORT 'B' Multichannel 96-127 Select Register - R/W

Reset = 0x0000 0000

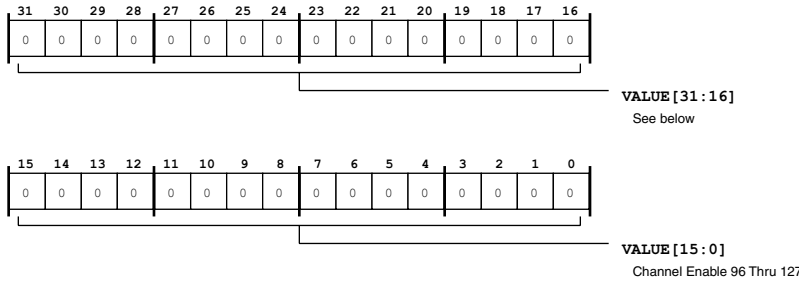


Figure 26-35: SPORT\_CS3\_B Register Diagram

**Table 26-36:** SPORT\_CS3\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 31:0    | VALUE    | Channel Enable 96 Thru 127. |

## Half SPORT 'B' Error Register

The `SPORT_ERR_B` contains error status and error interrupt mask bits for SPORT half 'B', including error handling bits for the half SPORT's primary and secondary channels and frame sync. Detected errors are frame sync violations or buffer over/underflow conditions.

### SPORT\_ERR\_B: Half SPORT 'B' Error Register - R/W

Reset = 0x0000 0000

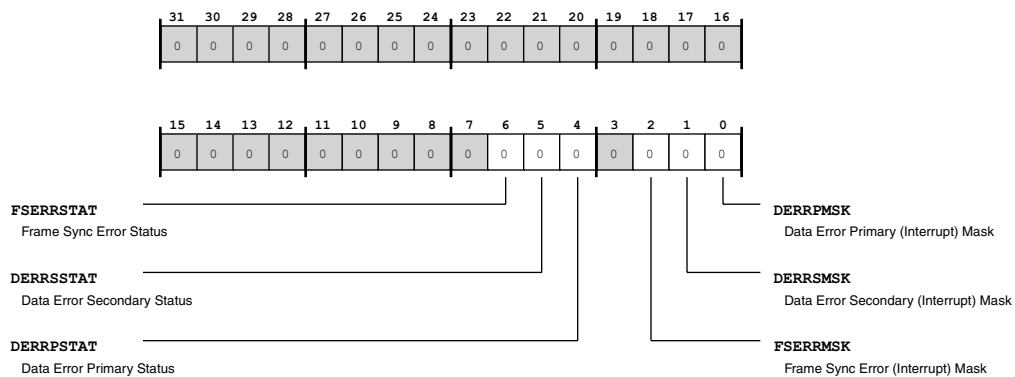


**Figure 26-36:** SPORT\_ERR\_B Register Diagram

**Table 26-37:** SPORT\_ERR\_B Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                |
|---------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| 6       | FSERRSTAT | Frame Sync Error Status.<br>The SPORT_ERR_B.FSERRSTAT bit indicates that the half SPORT has detected a frame sync when the bit count (bits remaining in the frame) is non-zero. When a half SPORT is receiving or transmitting, its bit count is set to a word length (for example, SPORT_CTL_B.SLEN = 32). After each serial clock edge, the half SPORT decrements the transfer's bit count. After the word is received or transmitted, the transfer's bit count reaches zero, and the half SPORT resets it (for example, to 32) on next frame sync. Normal SPORT data transfers always have a non-zero bit count value when active transmission or reception is occurring. Normal SPORT frame syncs occur during a zero bit count. |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | No error                                       |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Error (non-zero bit count at frame sync)       |
| 5       | DERRSSTAT | Data Error Secondary Status.<br>The SPORT_ERR_B.DERRSSTAT bit indicates the error status for the half SPORT's secondary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), SPORT_ERR_B.DERRSSTAT indicates transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), SPORT_ERR_B.DERRSSTAT indicates receive overflow status. This bit is used to clear the latch of SPORT status interrupt when triggered by secondary Data Error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRSEC status bit.                                                                                                                                                                                            |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | No error                                       |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Error (transmit underflow or receive overflow) |

**Table 26-37:** SPORT\_ERR\_B Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|
| 4       | DERRPSTAT | Data Error Primary Status.<br>The SPORT_ERR_B.DERRPSTAT bit indicates the error status for the half SPORT's primary channel data buffers. During transmit (SPORT_CTL_B.SPTRAN =1), SPORT_ERR_B.DERRPSTAT indicates transmit underflow status. During receive (SPORT_CTL_B.SPTRAN =0), SPORT_ERR_B.DERRPSTAT indicates receive overflow status. This bit is used to clear the latch of SPORT status interrupt when triggered by Primary Data Error. This bit can also be used to clear the read-only SPORT_CTL_B.DERRPRI status bit. |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | No error                                       |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Error (transmit underflow or receive overflow) |
| 2       | FSERRMSK  | Frame Sync Error (Interrupt) Mask.<br>The SPORT_ERR_B.FSERRMSK unmask (enables) the half SPORT to generate the frame sync error interrupt.                                                                                                                                                                                                                                                                                                                                                                                          |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Mask (disable)                                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Unmask (enable)                                |
| 1       | DERRSMSK  | Data Error Secondary (Interrupt) Mask.<br>The SPORT_ERR_B.DERRSMSK unmask (enables) the half SPORT to generate the data error interrupt for the secondary channel.                                                                                                                                                                                                                                                                                                                                                                  |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Mask (disable)                                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Unmask (enable)                                |
| 0       | DERRPMSK  | Data Error Primary (Interrupt) Mask.<br>The SPORT_ERR_B.DERRPMSK unmask (enables) the half SPORT to generate the data error interrupt for the primary channel.                                                                                                                                                                                                                                                                                                                                                                      |                                                |
|         |           | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Mask (disable)                                 |
|         |           | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Unmask (enable)                                |

## Half SPORT 'B' Multi-channel Status Register

The SPORT\_MSTAT\_B register indicates the current multi-channel being serviced among the half SPORT's active channels in multi-channel mode. The half SPORT increments the value by one in this register as each channel is serviced. The value in the SPORT\_MSTAT\_B register restarts at 0 at each frame sync.

SPORT\_MSTAT\_B: Half SPORT 'B' Multi-channel Status Register - R/NW

Reset = 0x0000 0000

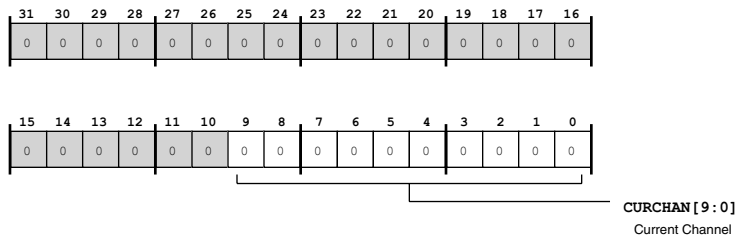


Figure 26-37: SPORT\_MSTAT\_B Register Diagram

Table 26-38: SPORT\_MSTAT\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 9:0     | CURCHAN  | Current Channel.<br>The SPORT_MSTAT_B.CURCHAN bits indicate the half SPORT's current channel being serviced in multi-channel mode. |

## Half SPORT 'B' Control 2 Register

The SPORT\_CTL2\_B register controls multiplexing options for sharing serial clock and frame sync signals across the related half SPORTs.

SPORT\_CTL2\_B: Half SPORT 'B' Control 2 Register - R/W

Reset = 0x0000 0000

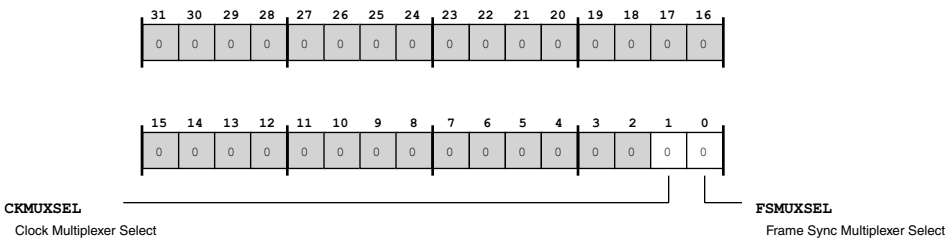


Figure 26-38: SPORT\_CTL2\_B Register Diagram

**Table 26-39: SPORT\_CTL2\_B Register Fields**

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | CKMUXSEL | Clock Multiplexer Select.<br>The SPORT_CTL2_B.CKMUXSEL bit enables multiplexing of the half SPORT' serial clock. In this mode, the serial clock of the related half SPORT is used instead of the half SPORT's own serial clock. For example, if SPORT_CTL2_B.CKMUXSEL is enabled, half SPORT 'B' uses SPORT_ACLK instead of SPORT_BCLK. |
|         |          | 0      Disable serial clock multiplexing                                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable serial clock multiplexing                                                                                                                                                                                                                                                                                                 |
| 0       | FSMUXSEL | Frame Sync Multiplexer Select.<br>The SPORT_CTL2_B.FSMUXSEL bit enables multiplexing of the half SPORT' frame sync. In this mode, the frame sync of the related half SPORT is used instead of the half SPORT's own frame sync. For example, if SPORT_CTL2_B.FSMUXSEL is enabled, half SPORT 'B' uses SPORT_AFS instead of SPORT_BFS.    |
|         |          | 0      Disable frame sync multiplexing                                                                                                                                                                                                                                                                                                  |
|         |          | 1      Enable frame sync multiplexing                                                                                                                                                                                                                                                                                                   |

## Half SPORT 'B' Tx Buffer (Primary) Register

The SPORT\_TXPRI\_B register buffers the half SPORT's primary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the primary channel. Either a program running on the processor core may load the data into the buffer (word-by-word process) or the DMA controller may automatically load the data into the buffer (DMA process).

The SPORT\_TXPRI\_B register acts as a three location buffer if SPORT data packing is disabled (SPORT\_CTL\_B.PACK=0); while it acts as two location buffer when packing is enabled (SPORT\_CTL\_B.PACK=1). So depending on PACK bit setting, two 32-bit words or three 32-bit words may be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT\_TXPRI\_B register contents are automatically loaded into the output shifter. The half SPORT may issue an interrupt (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt does not occur when the half SPORT is executing a DMA-based transfer, and the interrupt may be masked depending how its trigger output is configured in the TRU\_SSRn registers or TRU\_MTR register.

SPORT\_TXPRI\_B: Half SPORT 'B' Tx Buffer (Primary) Register - R/W

Reset = 0x0000 0000

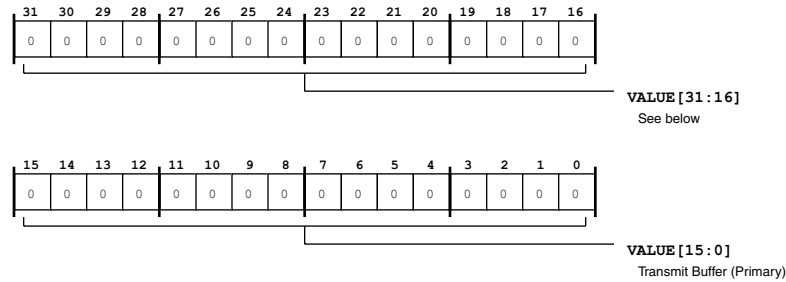


Figure 26-39: SPORT\_TXPRI\_B Register Diagram

Table 26-40: SPORT\_TXPRI\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Transmit Buffer (Primary).<br>The SPORT_TXPRI_B.VALUE bits hold the half SPORT's primary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions. |

Half SPORT 'B' Rx Buffer (Primary) Register

The SPORT\_RXPRI\_B register buffers the half SPORT's primary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the primary channel. When configured as a receiver, the half SPORT automatically loads the SPORT\_RXPRI\_B register from the receive shifter after a complete word has been received. The half SPORT then loads the data to internal memory through the DMA controller (DMA process) or through the SPORT process of a program running on the processor core (word-by-word process). With a data buffer and input shift register, SPORT\_RXPRI\_B register acts as a two-location buffer. Therefore SPORT can keep Two 32-bit received words at max in any one time (independent of SPORT\_CTL\_B.PACK bit setting).



**SPORT\_RXPRI\_B: Half SPORT 'B' Rx Buffer (Primary) Register - R/W**

Reset = 0x0000 0000

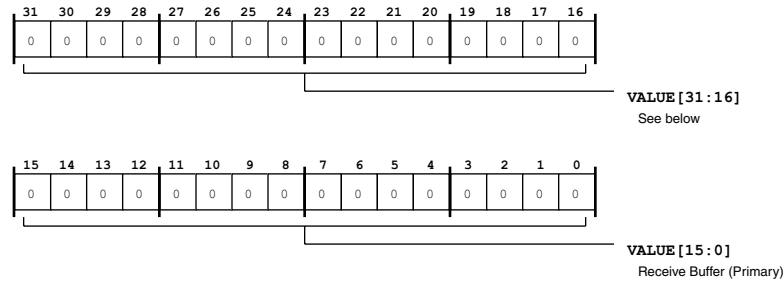


**Figure 26-40:** SPORT\_RXPRI\_B Register Diagram

**Table 26-41:** SPORT\_RXPRI\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Receive Buffer (Primary).<br>The SPORT_RXPRI_B.VALUE bits hold the half SPORT's primary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions. |

## Half SPORT 'B' Tx Buffer (Secondary) Register

The SPORT\_TXSEC\_B register buffers the half SPORT's secondary channel transmit data. This register must be loaded with the data to be transmitted if the half SPORT is configured to transmit on the secondary channel. Either a program running on the processor core may load the data into the buffer (word-by-word process) or the DMA controller may automatically load the data into the buffer (DMA process).

The SPORT\_TXSEC\_B register acts as a three location buffer if SPORT data packing is disabled (SPORT\_CTL\_B.PACK =0); while it acts as two location buffer when packing is enabled (SPORT\_CTL\_B.PACK =1). So depending on PACK bit setting, two 32-bit words or three 32-bit words may be stored in the transmit queue at any time. When the transmit register is loaded and any previous word has been transmitted, the SPORT\_TXSEC\_B register contents are automatically loaded into the output shifter. The half SPORT may issue an interrupt (transmit buffer is not full) when it has loaded the output transmit shifter, signifying that the transmit buffer is ready to accept the next word. This interrupt does not occur when the half SPORT is executing a DMA-based transfer, and the interrupt may be masked depending how its trigger output is configured in the TRU\_SSRn registers or TRU\_MTR register.

SPORT\_TXSEC\_B: Half SPORT 'B' Tx Buffer (Secondary) Register - R/W

Reset = 0x0000 0000

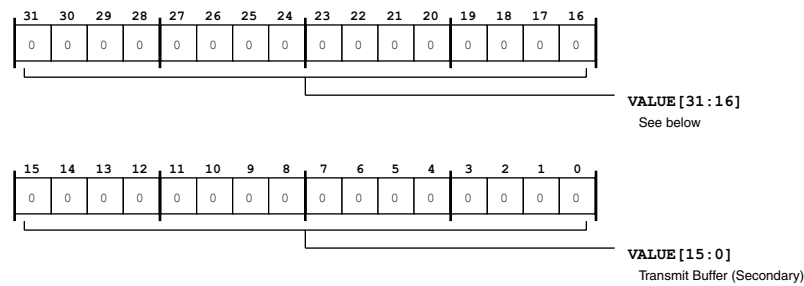


Figure 26-41: SPORT\_TXSEC\_B Register Diagram

Table 26-42: SPORT\_TXSEC\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Transmit Buffer (Secondary).<br>The SPORT_TXSEC_B.VALUE bits hold the half SPORT's secondary channel transmit data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions. |

Half SPORT 'B' Rx Buffer (Secondary) Register

The SPORT\_RXSEC\_B register buffers the half SPORT's secondary channel receive data. This buffer becomes active when the half SPORT is configured to receive data on the secondary channel. When configured as a receiver, the half SPORT automatically loads the SPORT\_RXSEC\_B register from the receive shifter after a complete word has been received. The half SPORT then loads the data to internal memory through the DMA controller (DMA process) or through the SPORT process of a program running on the processor core (word-by-word process). With a data buffer and input shift register, SPORT\_RXSEC\_B register acts as a two-location buffer. Therefore SPORT can keep Two 32-bit received words at max in any one time (independent of SPORT\_CTL\_B.PACK bit setting).

**SPORT\_RXSEC\_B: Half SPORT 'B' Rx Buffer (Secondary) Register - R/W**

Reset = 0x0000 0000

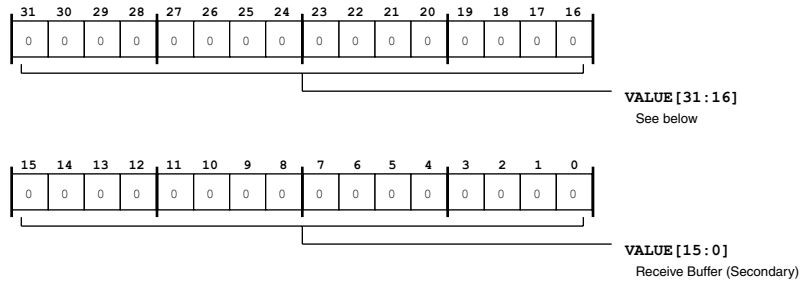


**Figure 26-42:** SPORT\_RXSEC\_B Register Diagram

**Table 26-43:** SPORT\_RXSEC\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Receive Buffer (Secondary).<br>The SPORT_RXSEC_B.VALUE bits hold the half SPORT's secondary channel receive data. Note that changes to the half SPORT operation mode (for example, toggling the SPORT_MCTL_B.MCE) empty the contents of this data buffer. For more information, see the SPORT_CTL_B and SPORT_MCTL_B register descriptions. |



## 27 ADC Control Module (ACM)

The processor includes an ADC control module (ACM) that provides an interface that synchronizes the controls between the processor and an Analog-to-Digital Converter (ADC). The analog-to-digital conversions are initiated by the processor, based on either external or internal events.

Traditionally, ADC sampling uses processor interrupts (initiated by the events) and the interrupt service routine programming of the appropriate peripheral (usually SPORT or SPI) for initiating the ADC conversion process. This traditional approach has some limiting factors:

- The ADC sampling instances are not precisely controlled due to interrupt latencies (which can vary) or due to variable instruction execution cycles
- Consumption of processor MIPS can be prohibitive, especially for high frequency of conversion related events.
- If the ADC requires any control signals (such as channel select pins, ADC mode select, ADC Range pin) with some specified Set-up, Hold or Zero time requirements with respect to sampling time, then providing such signals with GP flags may be difficult to implement in the application.

The ADC control module (ACM) answers the limitations of the traditional approach to sampling by providing a dedicated hardware, which samples the events and provides sampling signals with required timings to the ADC in real-time. It permits flexible scheduling of sampling instants and provides precise sampling signals to the ADC. The ACM approach both saves processor bandwidth and provides precise control for ADC sampling time. Furthermore, the processor can be interfaced directly to many ADCs without any glue logic required.

The ACM synchronizes the ADC conversion process (by providing ADC clock, ADC conversion start signal, and related ADC controls), but the actual data acquisition from the ADC is accomplished by other peripheral such as SPORT. On the ADSP-BF60x processors, the ADC module can be used in conjunction with either halves of SPORT1. The processor does not support ACM operation with the SPI. The following figure shows how an ADC can be interfaced using ACM and SPORT peripherals of processor.

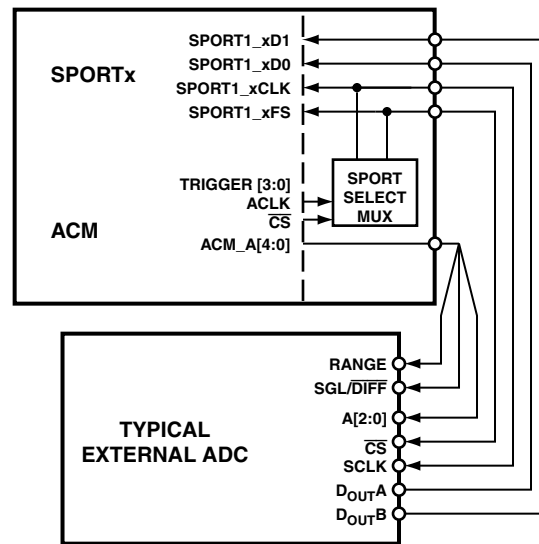


**Figure 27-1:** ADC/SPORT Interface

The ADSP-BF60x processors do not incorporate on-chip internal ADC.

## ACM Features

The ADC Control Module (ACM) offers following features and capabilities:

- Provides serial clock, chip select and five general-purpose control lines to control the ADC operations. Internally routes clock and CS signals to selected serial port.
- ACM can accept four trigger inputs, based on which it can precisely initiate the ADC sampling events. The Trigger inputs may be internally generated or externally supplied. Further, polarity of trigger inputs is configurable.
- ACM can handle 16 ADC sampling events per valid trigger received. Each event can be independently programmed to specify when to initiate ADC sampling with respect to trigger input.
- Two independent 32-bit ACM Timers can be used to divide 16 events into two groups of 8 events each.
- Automatically stops the ACM Timer after completion of associated events in order to save the power.
- Four-deep pending FIFO to queue the active events when ACM is busy.
- ACM can internally generate serial clock up to  $SCLK/2$  rate. Improved granularity for internal clock generation, allowing both odd and even  $SCLK:ACLK$  ratios.
- ACM clock can be gated (active only during enabled events) to interface it with SPI-compatible ADC's.
- When initiating ADC sampling cycle, the width of chip select signal-which can be also used by ADC as start of conversion-, can be configured from 1  $ACLK$  to 256  $ACLK$ s. Further the polarity of this signal can be configured as active-high or active-low signal.

- Auto ACLK adjustment at the time of CS assertion. After assertion of CS signal, the first edge of ACLK can be configured to be either rising edge or falling edge.
- The five general-purpose control lines provided by ACM can be programmed for required Set-up and Hold time with respect to ADC sampling cycle. Additionally, Zero time can be inserted between two successive sampling cycles.
- ACM provides 16 Event Order registers (one per each event) which indicates the order at which events were handled. Optionally these registers can be automatically cleared by ACM's trigger input.
- The ACM hardware flags the appropriate event completion status bit upon completion of an event. If an event got missed, appropriate event missed status bit is flagged. Each event has separate bits. Optionally Event Completion Interrupt and Event Missed Interrupt can be triggered upon these respective conditions.
- Predictable latency between the internal occurrence of an event and the assertion of a sampling event.
- ACM can operate as Trigger Master to provide signal to TRU upon completion of events.

## ACM Functional Description

The ADC Control Module uses internal ACM timers and the event time register to create events. The user has to enable one of the timer (or both timers) for the ACM operation. Appropriate event control register and event time register values also have to be programmed. After receiving valid trigger on selected trigger input, the timer starts counting. If at anytime the timer count matches with the time specified in the event time register (ACM\_ETx) of an enabled event (associated with that timer), the comparators generate an active event signal to the timing generation unit to start the ADC access. The counter continues counting, and for each matching with enabled event time, the ACM gives an event signal to the timing generation unit.

Figure shows the ACM operation where only two events (Event0 and Event3) are enabled. The line labeled “ADC Controls” depicts the timing of the ADC control signals: A[4:0].

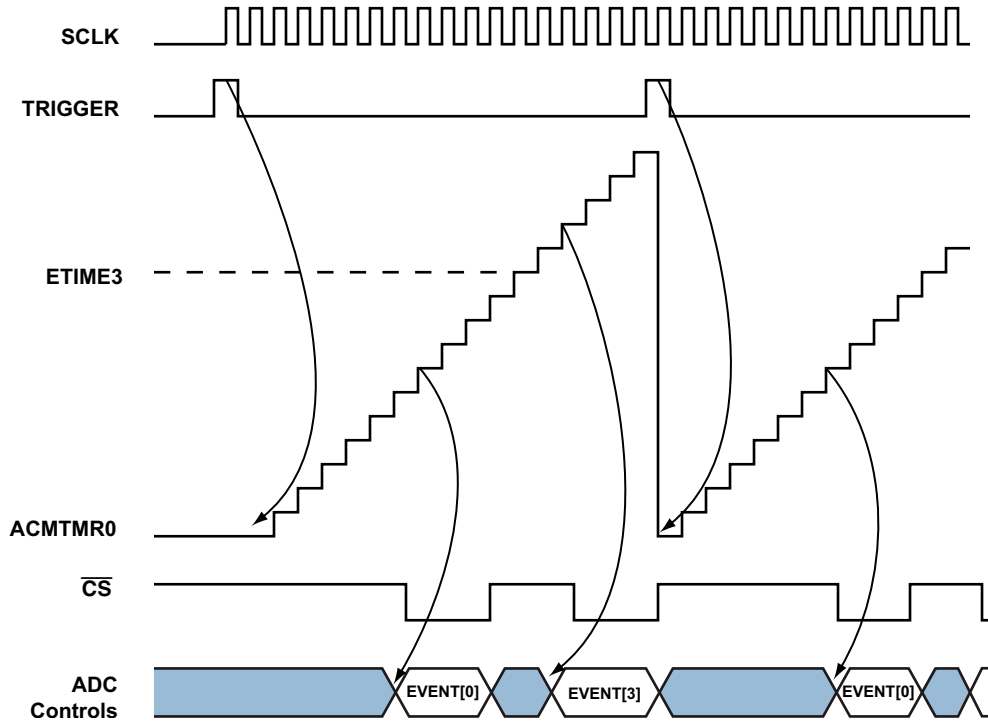


**Figure 27-2:** ACM Operation, Two Events

Note that, this figure depicts a usage case in which ACM Event3 Time register is programmed with a count value that is greater than that programmed in Event0. So, Event3 occurs after Event0. There are, however, no restrictions on the order of the different events. Event0 time can be greater than, less than or equal to Event3 time.

If value in Event0 Time register is less than the value in Event3 Time, Event3 occurs after Event0; while if the value in Event0 is greater than the value in Event3, then Event3 occurs before Event0. Whereas if Event0 time is equal to the Event3 time, the events are processed according to their priorities. Only the highest priority event is processed, and other lower priority event is missed (even if there is a space in pending event FIFO), as both events, are handled by same ACM Timer. Event0 has the highest priority. So in this case, the timing generation unit processes Event0, while Event3 is missed. In this case, the Event3 missed status bit (EM3) is set in the missed event status register (ACM\_MEVSTAT), and the Event Missed bit (EMISS) is set in the ACM Status (ACM\_STAT) register, indicating that an event has been missed.

If event times are not sufficiently spaced apart, an event could occur while a previous event is underway (while the CS of the previous event is asserted). In this situation, the second event is queued in the pending event FIFO. If the pending event FIFO is full, the event will not get queued and will be missed. This can happen mostly when enabling both ACM Timers with different trigger inputs and the sources of these triggers are not synchronized together. In this case, it is possible that the events controlled by the two timers to overlap. It is therefore important to consider the possibility of events occurring either simultaneously or being missed when enabling events on two asynchronously-triggered timers. It is programmer's responsibility to ensure that the values in the event time registers do not lead to event misses. The appropriate event miss bit in the ACM\_MEVSTAT and ACM\_STAT registers will be flagged, upon missing any event.



When both ACM timers are enabled, and if they triggered the events simultaneously, the event triggered by ACMTMR0 is given higher priority. For example, when an ACMTMR0 event (one of events 0 through 7) and ACMTMR1 event (one of events 8 through 15) occur simultaneously, the ACMTMR0 event is processed by the timing generation unit or is queued in the pending event FIFO before the processing or the queuing of the ACMTMR1 event.

When all the events enabled for a given ACM timer (ACMTMR<sub>x</sub>) are processed, the ACM timer stops incrementing. (Note that this timer action is not reflected in Figure). And corresponding Event Completion bit (ECOMP<sub>x</sub>) in the ACM Status register is flagged. The same bit also reflects in Event Completion Status register (ACM\_EVSTAT), which can optionally generate Event Completion interrupt if the corresponding bit in the Event Completion Interrupt mask register (ACM\_EVMSK) is set. Two separate bits are available, one for each ACM Timer.

The ACM can be used to generate various sequences of ADC sampling events through appropriate programming of event time registers, event control registers, and triggers. For more information, see the usage cases described in [Emulation Mode Use Case](#).

## ACM Event Handling Latency

The ACM ensures a predictable latency between the internal occurrence of an event (event time value matching the ACM timer count value) and the assertion of a sampling event by the timing generation unit (the assertion of CS and other ACM signals as appropriate).

Latency between occurrence of Event to CS assertion = ( $t_S + t_{ED}$ ) SCLK1 cycles, where:

- $t_S$  = ADC control setup cycles programmed in ACM\_TCO register
- $t_{ED}$  = 1 SCLK cycle latency

This predictable latency is applicable only when events are generated when the Timing Generation Unit is idle. If this Timing Generation Unit was processing a prior sampling event, then the new event is held in the Pending Event FIFO, and the latency is increase by the duration that the new event is held in the pending event FIFO.

If an external trigger input is selected as a trigger input of the ACM, then synchronization to this external signal leads to a 3 SCLK cycle fixed delay and 1 SCLK cycle variability due to delays in latching asynchronous external triggers. When the external trigger is synchronous to SCLK, the 1 SCLK cycle variability is eliminated and the latency from the external trigger to the start of the count of an ACM timer becomes fixed at 3 SCLK cycles. This latency is denoted as  $t_{TRIG}$ .

As a result, the total latency between an external trigger and the assertion of an ADC sampling event, assuming that the sampling event does is not queued in the pending event FIFO, is:

$$\text{Total Latency} = t_{TRIG} + t_{ED} + t_{PD} + t_S$$

The following figure shows latency details from the occurrence of external triggers to the assertion of ADC sampling events.

In the following figure, observe the following timing definitions:

- $t_{TRIG}$  = trigger to timer start delay (3 to 4 SCLK)
- $t_{PD}$  = Event Time (programmed in ACM\_ETx register of event).
- $t_{ED}$  = internal event delay (1 SCLK)
- $t_S$  = set up time (programmed in ACM\_TC0 register)
- $t_{CSW}$  = CS width (programmed in ACM\_TC0 register)
- $t_H$  = hold time (programmed in ACM\_TC1 register)

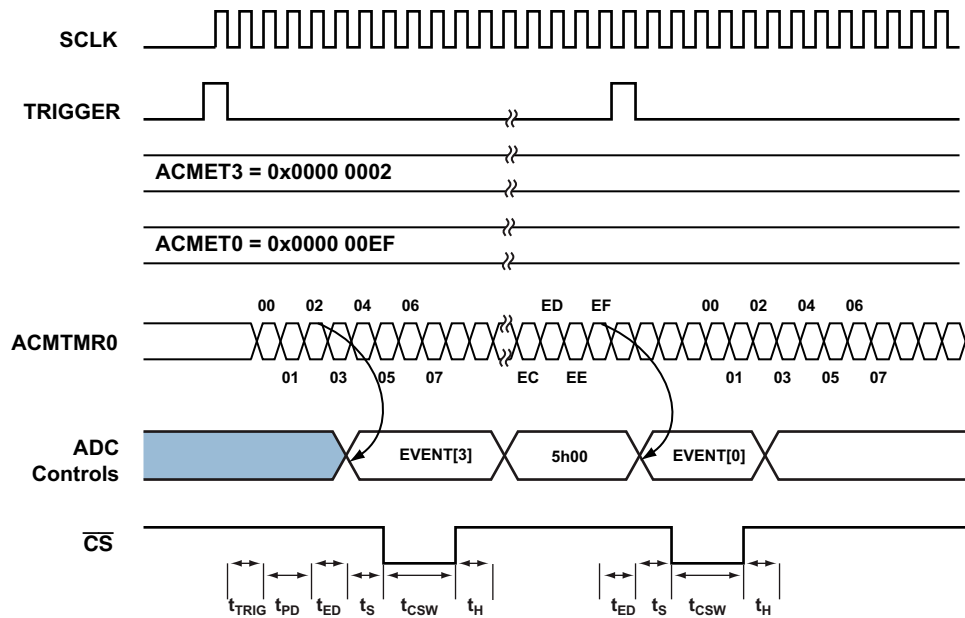


**Figure 27-3:** Latency of External Triggers to Assertion of ADC Sampling Events

## ACM Timing Specifications

The AC timing of the ACM signals are specified in the ADSP-BF60x Embedded Processor Data Sheet. When trigger sources external to the processor are used for triggering the ACM (for example, external signals on the GPIO, timer or PWM sync pins), the minimum pulse-width for such trigger sources needs to be greater than one SCLK period in order to detect it as a valid trigger by the ACM trigger logic.

- When the ACM is used in conjunction with the SPORT, the setup and hold timing requirements for the SPORT data signals with respect to ACLK are different from those requirements with respect to

internally-generated or externally-supplied SPORT clock. Consult the ADSP-BF60x Embedded Processor Data Sheet for information on these timing requirements.

When using Gated clock mode (`ACM_CTL.CLKMOD=1`), the interfaced serial mode should also be configured in Gated clock mode (`SPORT_CTL.GCLKEN=1`). In this case, it is required to satisfy some conditions in order to set-up the serial port in Gated clock mode. These conditions are:

- The serial port needs at least 7 serial clock cycles between enabling the SPORT and first frame sync. If this requirement is not met, the SPORT may drop the first data. (For subsequent data this requirement is not applicable).
- The frame sync should be in the inactive (de-asserted) state when the SPORT is enabled. Otherwise one extra cycle (in addition to the above mentioned) is needed before the frame sync can be applied. If this requirement is not met, the SPORT may drop the first data.

## ACM External Pin Timing

The ACM clock (ACLK) is derived internally from SCLK1 clock using the CKDIV divider specified in the ACM\_TC0 register. The other output signals such as ADC control pins (`ACM_A[4:0]`) and CS are driven on the rising edge of SCLK1 clock. As a result, these signals may not be synchronous to ACLK. The setup, hold, and other timing parameters of the ADC control signals, width of CS signal and the frequency of ACLK can be configured in the ACM timing configuration registers (`ACM_TCx`). The polarity of CS and ACLK can be configured in the ACM control register (`ACM_CTL`). The timing parameters of the ADC control pins (`ACM_A[4:0]`) cannot be individually specified.

The inactive period of CS ( $t_{CSIW}$  as shown in the following figure) is the sum of the three timing parameters – Setup Time ( $t_s$ ), Zero Time ( $t_z$ ) and the Hold Time ( $t_H$ ):

$$t_{CSIW} = t_s + t_z + t_H.$$

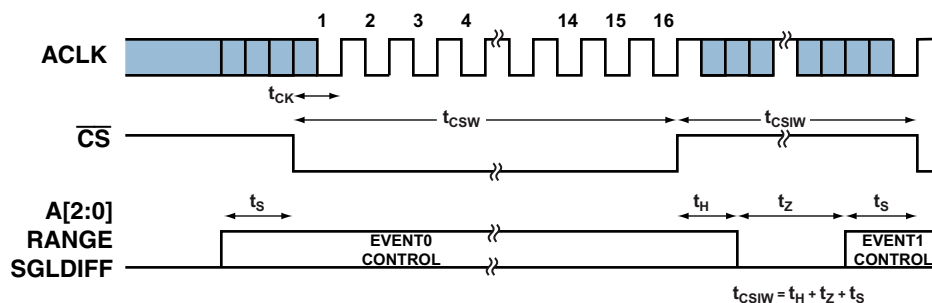


**Figure 27-4:** ACLK Timing Reference

Appropriate specification of values of these three parameters can yield the desired inactive period of CS.

In order to provide a predictable latency from the occurrence of an internal event to the assertion of an external ADC sampling event, the ADC controls and CS must be driven on the rising edge of SCLK. Therefore, the Setup Time ( $T_s$ ) of these signals is specified in terms of SCLK. However, the hold-time and zero-time are specified in terms of ACLK cycles.

To achieve accurate timing relationship between CS and ACLK (which is a normally a free running clock), the ACLK signal is re-aligned with the active edge of CS. This realignment of ACLK ensures that the setup time of the first active edge of ACLK, with respect to the active edge of CS, is at least 1ACLK cycle.

The figures in the following sections show various scenarios of ACLK re-alignment. All of these figures assume an ACLK:SCLK ratio of 1:4.

- Case 1—Chip Select Asserted during the High Phase of ACLK
- Case 2—Chip Select Asserted During the Low Phase of ACLK
- Case 3—Chip Select Asserted Right Before the Falling Edge of ACLK
- Case 4—Chip Select Asserted Right Before the Rising Edge of ACLK
- Case 5—ACLK Polarity Set to 1 (CLKPOL=1)

Notice that re-alignment of ACM clock causes suppression or extension of clock edges, leading to duty cycle variation. It is important to ensure that systems interfacing with the ACM can tolerate such duty cycle variation.

The figures show both the ACM-generated CS signal, which is output externally onto the appropriate ACM0FS pin, and the serial port receive frame sync (SPORT1\_xFS) signal, which is an internal signal that is routed to the frame sync input of the appropriate SPORT.

Please note that ACM clock polarity can be configured using the CLKPOL bit of ACM control register. After assertion of CS signal, the first edge of ACLK can be configured to be either rising edge or falling edge. Also, by default the clock is free running; it is possible to gate the ACM clock during inactive CS period using the CLKMOD bit of the same register.

## **Case 1—Chip Select Asserted During the High Phase of ACLK (CLKPOL=0)**

The following figure shows the realignment of ACLK when CS is asserted during the high phase of ACLK. The first edge of ACLK after the assertion of CS is the falling edge.

The two reference clock signals (RefACLK1 and RefACLK2) are shown to illustrate how the ACLK signal can be generated from a free running clock (RefACLK1) in order to meet the timing requirements between ACLK and CS. RefACLK2 is based on the free running clock RefACLK1, but is adjusted such that its period is immediately reset upon the assertion of CS. The resulting ACLK signal, shown in the figure, is such that the time from the active edge of CS to the falling edge of ACLK is constant at a period of 1ACLK cycle.

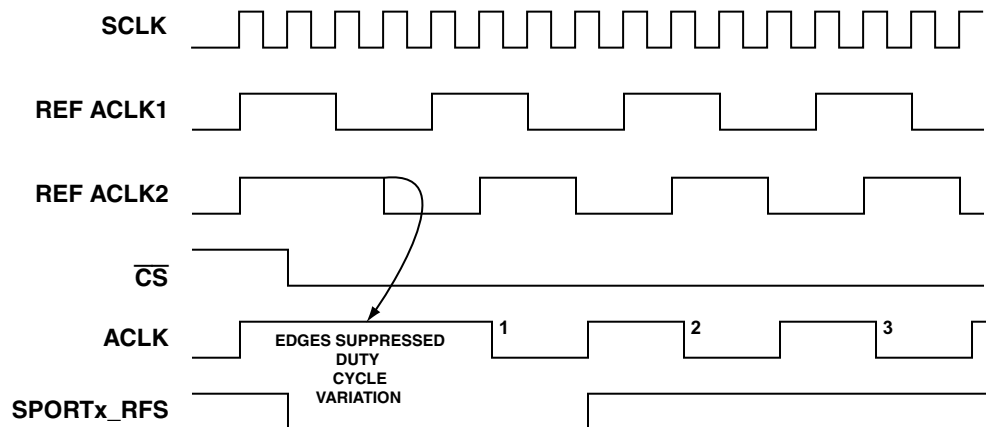


**Figure 27-5:** Chip Select Asserted During the High Phase of ACLK

### Case 2—Chip Select Asserted During the Low Phase of ACLK (CLKPOL=0)

When CS is asserted during the low phase of ACLK, as shown in the following figure, ACLK is immediately pulled high causing a duty cycle variation. In this case, similar to Case 1, the time from the active edge of CS to the falling edge of ACLK is 1ACLK period.

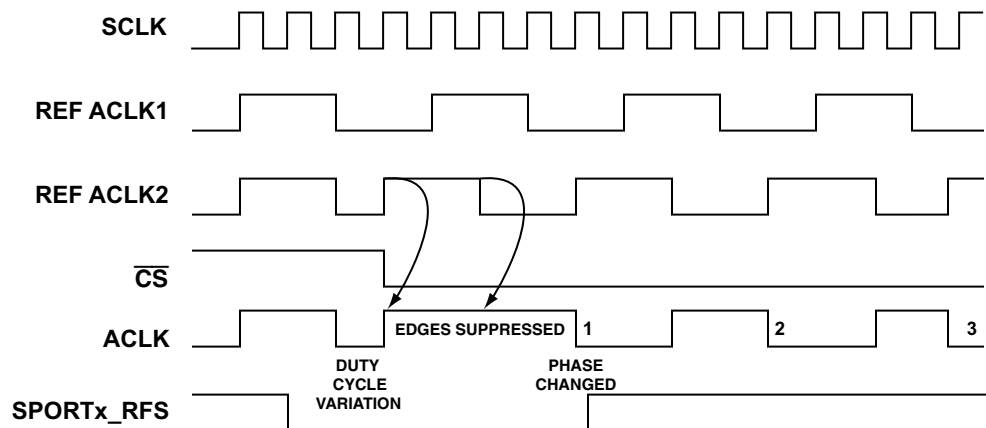


**Figure 27-6:** Chip Select Asserted During the Low Phase of ACLK

### Case 3—Chip Select Asserted Right Before the Falling Edge of ACLK (CLKPOL=1)

When CS is asserted right before the falling edge of ACLK, the falling edge of ACLK is suppressed, as shown in the figure. This ensures that the time from the active edge of CS to the falling edge of ACLK is constant at a period of 1ACLK cycle.

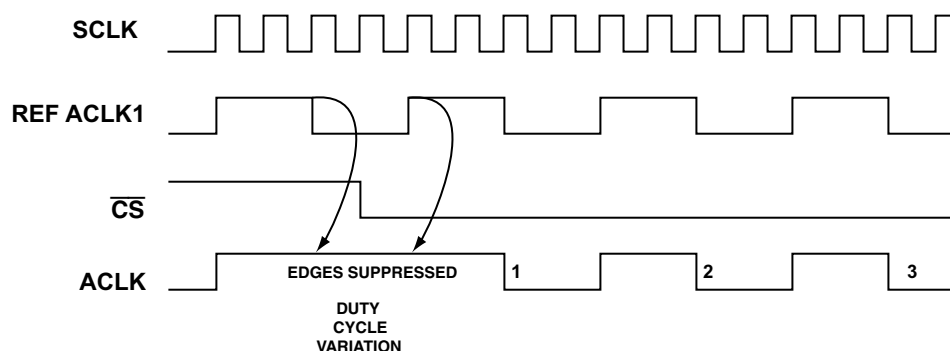


**Figure 27-7:** Chip Select Asserted Right Before the Falling Edge of ACLK

#### Case 4—Chip Select Asserted Right Before the Rising Edge of ACLK ( $CLKPOL=0$ )

When CS is asserted right before the rising edge of ACLK, the high phase of ACLK is extended, as shown in Figure. This extension ensures that the time from the active edge of CS to the falling edge of ACLK is constant at a period of 1 ACLK cycle.

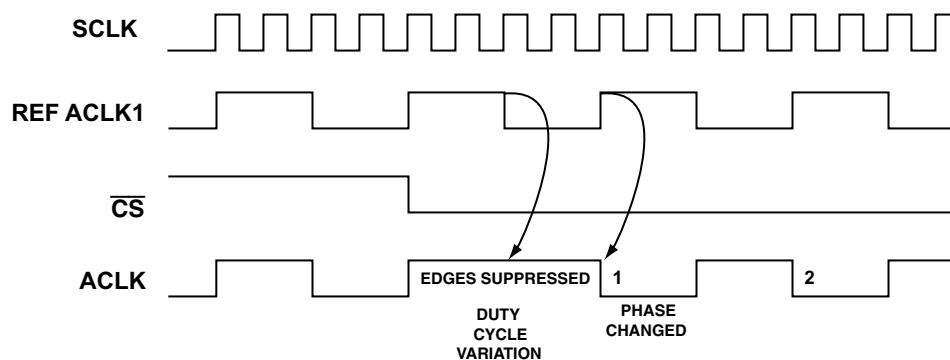


**Figure 27-8:** Chip Select Asserted Right Before the Rising Edge of ACLK

#### Case 5—ACLK Polarity Set to 1 ( $CLKPOL=1$ )

When the ACLK polarity is set to 1 (bitCLKPOL is set to 1 in the ACM Control register), the first ACLK edge after the assertion of CS is the rising edge. The ACM ensures that the time from the active edge of CS to the rising edge of ACLK has a constant duration of 1 ACLK cycle. The figure shows an example diagram of the case where CLKPOL=1.

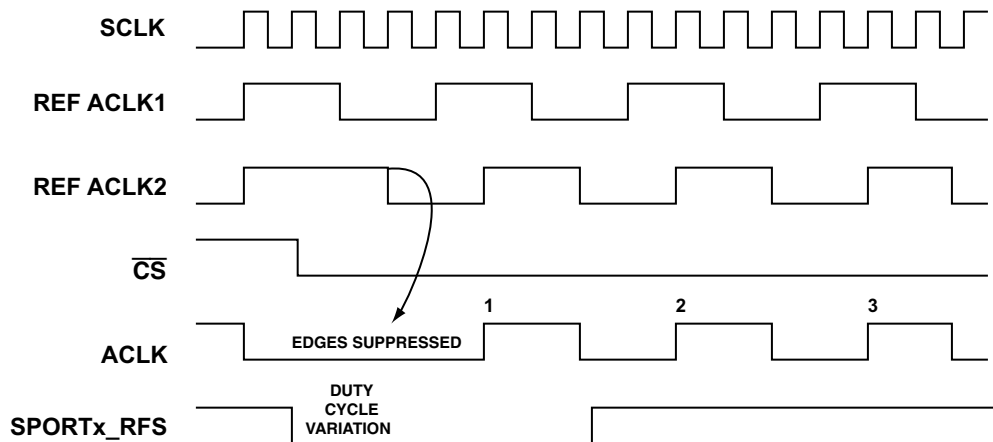


**Figure 27-9:** ACLK Polarity Set to 1

## ACM Architectural Concepts

The following sections provide information on the architecture of the ACM module.

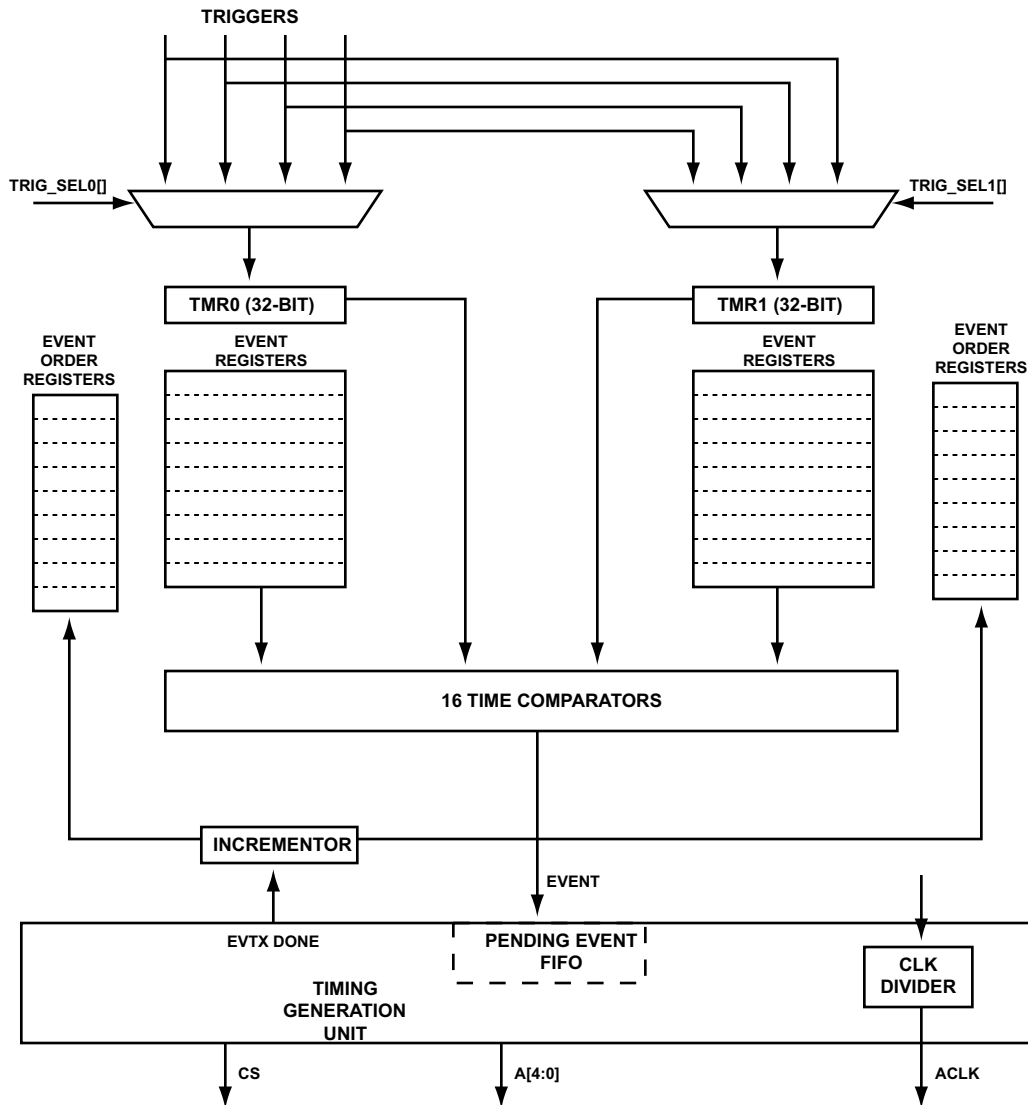
### ACM Block Diagram

The ADC Control Module consists of two independent 32-bit ACM Timers, 16 Event Register pairs, 16 Event Comparators and a Timing Generation Unit.

ACM can accept four trigger inputs (internal as well as external signals); and on receiving a valid trigger on selected trigger input, the ACM timer/s start counting (based on the Mode of ACM). The trigger input can be independently selected for each timer.

Two sets of 8 event register pairs (total 16 event register pairs) determines the ADC controls for each ADC sampling, and also determines when the sampling happens. The Event register pair consists of: Event Control Register (ACM\_ERx) and the Event Time Register (ACM\_ETx). The Event Comparators Unit, compares the ACM Timer's count with Event Time of associated enabled events and, upon matching the count, signals appropriately to Timing Generation Unit. The Timing Generation Unit starts handling the events by driving the CS and ACM[4:0] signals accordingly.

The ACM Block diagram is shown below:



The following sections discusses about these blocks in detail.

## ACM Trigger Inputs

ACM can accept four trigger inputs, based on which ACM timers start running at **SCLK1** rate. ACM contains two 32-bit internal timers; each can be independently configured to use one of these trigger input. The two separate fields, **TRGSEL0** and **TRGSEL1**, of ACM Control register selects the trigger input for **ACMTIMER0** and **ACMTIMER1** respectively.

Therefore, ACM uses two trigger inputs when both ACM timers are enabled for different trigger inputs. However, it uses only one trigger if both ACM Timers are enabled for same trigger input or if single ACM Timer is enabled. The non-selected trigger inputs are 'don't care' for ACM. So, at most two and at least one selected trigger input should be active in the system for ACM to start its operation.



The four possible trigger inputs are briefly described below:

- ACMTriggerinput0 (ACM\_T0) – PE8:

Trigger input- 0 is sourced from PE8 pin of the PORTE (sometimes also referred as GPIO[72]). When ACM is enabled, input tap on PE8 pin is enabled; so, ACM trigger input can be from any source (internal or external) depending on PE8 pin configuration.

When PE8 pin is configured in GPIO mode (FER=0), the source of GPIO signal may be either internal or external depending on the GPIO direction configured in PORTE\_DIR register. When PE8 pin is configured in function mode (FER=1), the trigger- 0 input is sourced in from peripheral signals based on PORTE\_MUX register setting for PE8 pin. For example the ACM can source this trigger input from PWM0\_SYNC signal (internally generated by PWM unit or externally provided), if MUX bits for PE8 pin are set to b#00. Whereas, if MUX bits for PE8 are configured to b#01, trigger input is sourced from PPIO\_FS1 signal (internally generated by EPPIO unit or externally-generated).

So, FER and MUX bits for PE8 pin must be programmed appropriately considering the source of trigger input. Enabling input tap ensures that ACM operation would not interfere with module driving the PE8 pin.

- ACMTriggerinput1 (ACM\_T1) – PG5:

Trigger input- 1 is sourced from PG5 pin of the PORTG (sometimes also referred as GPIO[101]). When ACM is enabled, input tap on PG5 pin is enabled; so, ACM trigger input can be from any source (internal or external) depending on PG5 pin configuration.

When PG5 pin is configured in GPIO mode (FER=0), the source of GPIO signal may be either internal or external depending on the GPIO direction configured in PORTG\_DIR register. When PG5 pin is configured in function mode (FER=1), the trigger- 1 input can be sourced in from peripheral signals based on PORTG\_MUX register setting for PG5 pin. E.g. ACM can source this trigger input from PWM1\_SYNC signal (internally generated by PWM unit or externally generated), when MUX bits for PG5 pin are set to b#10.

So, FER and MUX bits for PG5 pin must be programmed appropriately considering the source of trigger input. Enabling input tap ensures that ACM operation would not interfere with module driving the PG5 pin.

- ACMTriggerinput2 (ACM\_T2) – ACMSlaveTriggerID33:

The Trigger Routing Unit of processor provides system-level sequence control without core intervention. When this trigger input mode is selected, ACM acts as Trigger Slave and accepts trigger through Trigger Slave ID- 33. The 'Slave Select' (SSR) field of TRU\_SSR33 register should be configured to receive triggers from a specific trigger Master. In this way, ACM\_T2 trigger input can accept triggers asserted by that particular trigger Master or through software by writing ID of that trigger Master to one of the four fields in the TRU\_MTR register. The trigger response from selected master is internally routed to ACM trigger input. This way, ACM slave trigger ID- 33 is capable of receiving any one of the 86 internal triggers available.

- ACMTriggerinput3 (ACM\_T3) – A CMSlaveTriggerID34:

Similar to ACM\_T2 input, ACMTriggerInput3 (ACM\_T3) is internally provided by Trigger Routing Unit of the processor. When this trigger input mode is selected, ACM acts as Trigger Slave and accepts trigger through Trigger Slave ID- 34. The 'Slave Select' (SSR) field of TRU\_SSR34 register should be configured to receive triggers from a specific trigger Master. In this way, ACM slave trigger ID 34 can accept triggers asserted by that particular trigger Master or through software by writing ID of that trigger Master to one of the four fields in the TRU\_MTR register. The trigger response from selected master is internally routed to ACM trigger input. This way, ACM slave trigger ID 34 is capable of receiving any one of the 86 internal triggers available.

Refer Trigger Routing Unit chapter for more details about Trigger slaves and Trigger masters.

For all trigger input signals, the active edge of the trigger is programmable in the ACM Control register as either rising edge or falling edge trigger.

The following figure shows the detailed ACM trigger generation logic.

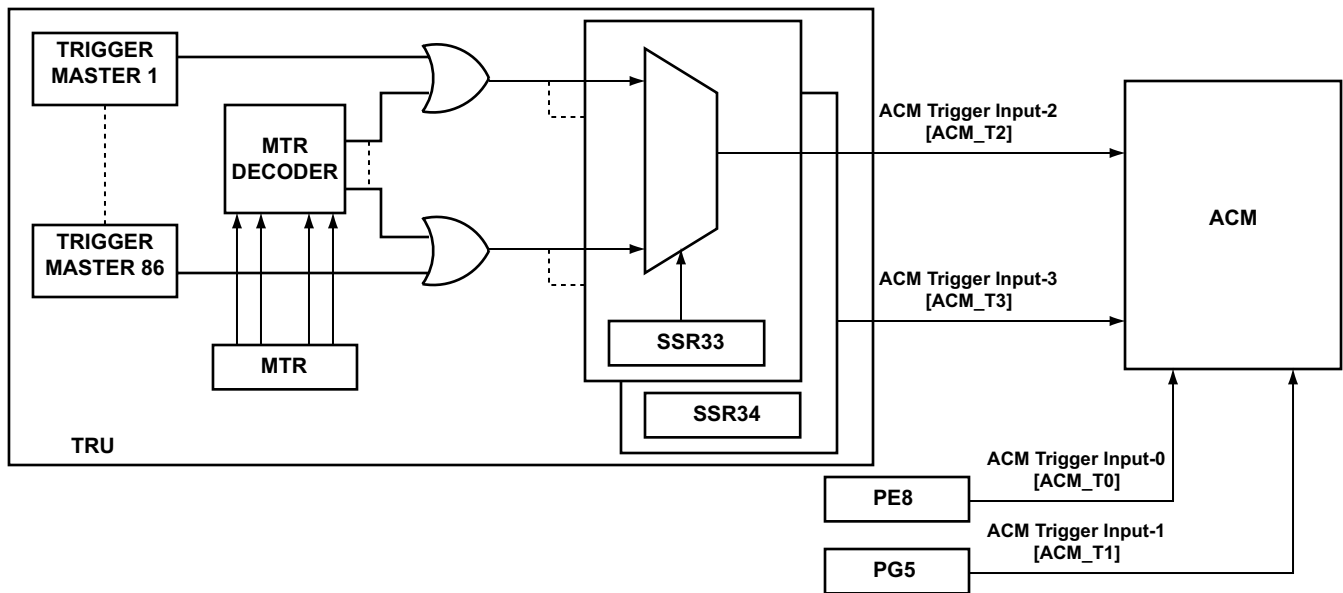


**Figure 27-10:** Detailed ACM Trigger Generation Logic

When trigger sources external to the processor (e.g. ACM\_T[1:0]) are used for triggering the ACM Timers, the minimum pulse width for such trigger sources needs to be greater than one SCLK1 period.

**NOTE:** A latency of no more than four SCLK1 cycles exists between external trigger and ACM Timer starts counting. Please refer to [ACM Event Handling Latency](#) section for further details.

## ACM Timers

The ACM module has two independent 32-bit timers (ACMTMR0 and ACMTMR1) which start counting at system clock (SCLK1) rate upon detecting valid trigger on selected trigger input. The timers can be independently enabled using the timer enable bits (ACMTMR0EN, ACMTMR1EN) in the ACM Control register; and

can be independently configured for one of the four trigger inputs with configurable polarity of the signal. At least one ACM timer should be enabled for proper ACM operation.

By default, each ACM timer has 8 event associated with it. If both timers are enabled, the Event[7:0] are associated with ACMTMR0; while the Event[15:8] are associated with ACMTMR1. However, if only one timer is enabled, all of the event registers are associated with that particular timer. For example, if only ACMTMR1 is enabled, (if ACMTMR0EN=0 and ACMTMR1EN=1), all 16 events, Event[15:0], will be handled by ACMTMR1.

These timers start counting when a trigger input occurs that is selected for that particular timer. The timer only stops counting under one of the following conditions:

1. A timer rollover occurs.
2. All the events associated with the trigger have completed.

Note that a timer rollover can never happen, unless the Event Time Register of an event is programmed at some point after the trigger occurs; this is a practice that is contrary to ACM programming guidelines.

In the second case, the exact time at which the Timer stops counting depends on the FIFO state when the last event occurred internally.

If a trigger occurs while the timer is counting, the time resets and starts counting again. In this case, some of the events may miss resulting in flagging appropriate status bit and optionally an Event Missed interrupt.

When an ACM timer is disabled or the ACM itself is disabled, the timer resets to zero.

## **Event Register Pairs**

An Event, for the ACM, is a point in time where ADC sampling has to happen on a particular channel of the ADC with the specified control settings of the ADC.

ACM can handle total 16 events which are grouped into two sets of 8 events. As explained in ACM Timer sections, either 8 events can be assigned to each of the timers (if both timers are enabled) or all 16 events can be assigned to one particular timer (if only one timer is enabled).

All events can be independently configured and enabled. The enabled events determine the ADC controls and timing for each ADC sampling interval. Each event consist a register pair comprising an event control register (ACM\_ERx) and an event time register (ACM\_ETx). The ACM\_ERx register enables a particular event and determines settings for the ADC control lines (ACM\_A[4:0]) for that particular ADC conversion. The ACM\_ETx register determines the time offset from the corresponding ACM timer trigger input to the start of that particular event (i.e. when the event has to occur w.r.t. trigger input). This time offset should be specified in terms of system clock 1 of the processor.

At least one event, associated with enabled ACM Timer, should be enabled for ACM to execute ADC sampling.

## Event Comparators Unit

The Event Comparator block consists of 16 event time comparators which determine when an enabled event should happen. After detecting valid trigger on selected trigger input, ACM timer starts running at system clock 1 rate. The comparators compare the ACM timer count with the event time specified in the `ACM_ETx` register of the enabled event. If the time value matches, the comparators indicate an active event signal to the timing generation unit.

If an event happens when another event is ongoing, the occurred event is stored in the pending event FIFO of Timing Generation Unit. If more than one event associated with a ACM Timer are active during the same `SCLK` cycle, only the highest priority event is processed, and all other events are missed (even if there was space in the pending event FIFO). However, if both Timers are enabled and if multiple events associated with both ACM Timers are active at the same `SCLK` cycle, then two events, one highest priority active event per each timer, are signaled.

The priority of events is fixed; event with lowest event ID has higher priority compared other events.

When both ACM Timers are enabled, Event0 has the highest priority and Event7 has the lowest priority in the `Event[0:7]` group associated with `ACMTIMER0`. Similarly Event8 has the highest priority and Event15 has the lowest priority in the `Event[8:15]` group associated with `ACMTIMER1`. So if Event1 and Event5 occur simultaneously, then Event5 is missed, even if there is space in the pending FIFO. But between the `Event[0:7]` and `Event[8:15]` group, simultaneous events can be written into the FIFO. For example, if Event0 and Event9 occur together, then both are written into the FIFO, `Event[0:7]` group has higher priority, so the order of Events in the FIFO is Event0 first and then Event9. If Event1, Event5, Event9, Event15 happened together, then Event5, and Event15 are missed and Event1 and Event9 are put into the FIFO (Event1 first followed by Event9). When events that are triggered by both timers occur simultaneously, the event triggered by `ACMTMR0` is given higher priority.

When only single ACM Timer is enabled, then all 16 events, `Event[0:15]`, are assigned to that timer. So, Event0 has highest priority; while Event15 has lowest priority. So, in this case, if Event1, Event5, Event9, Event15 happened together, then only Event1 will be placed in forwarded, while Event5, Event9 and Event15 will be missed, even if there is space in the pending FIFO.

If an event is missed, the `EMISS` bit in the ACM Status register (`ACM_STAT`) and the corresponding bit in the ACM Event Missed Status register (`ACM_MEVSTAT`) are set.

## Timing Generation Unit

After event signaling from event comparators, the timing generation unit initiates an ADC sampling interval as per settings of that particular event. It generates ADC control signals based on the Event parameter field of the `ACM_ERx` register setting. The timings of output signals (`ACLK`, `CS`, `A[4:0]`) are determined by the ACM Timing registers (`ACM_TCx`) which contains the fields like Set-up time, Hold time and Zero time for these signals in addition to ACM clock divider.

The Pending FIFO is part of Timing Generation Unit. If an event happens when ACM is busy with another event, the occurred event is stored in the pending event FIFO. This pending event is serviced (for example, the ACM starts an ADC conversion for the event that occurred), after completion of ongoing event.

The pending event FIFO has a depth of 4, so it can hold up to four pending events. If an event occurs when the pending event FIFO is full, that event is missed. If an event is missed, the `EMISS` bit is set in the `ACM_STAT` register and the corresponding bit in the `ACM_MEVSTAT` register also is set.

On disabling the ACM, all the pending entries in the pending FIFO are flushed.

## Status Flags and Interrupts

The ACM provides a read-only Status register (`ACM_STAT`) to check the module activities such as which event is currently being serviced, whether any event has been missed events or all the events has been serviced for the current trigger.

In addition to this Status register, ACM also provides two general-purpose status registers, ACM Event Completion Status register (`ACM_EVSTAT`) and ACM Missed Event Status register (`ACM_MEVSTAT`). The `ACM_EVSTAT` register specifies servicing of which enabled events has completed for a particular trigger cycle; while `ACM_MEVSTAT` register specifies which enabled event has been missed for that particular trigger cycle. This information is provided for all the 16 events via individual bits.

Based on these status bits, ACM can generate two interrupts, event completed or event missed, for each event. These interrupts can be selectively enabled for particular ACM events via ACM Completed Event Interrupt Mask Register (`ACM_EVMSK`) and ACM Missed Event Interrupt Mask Register (`ACM_MEVMSK`) registers.

The Event Completion interrupt is generated only after the entire event completes externally (for example, when `CS` signal goes inactive, and Hold Time (`TH`), Zero Time (`TZ`) periods are completed for that particular event). The `ACM_EVSTAT` register provides the status of each event indicating which event has caused the interrupt. It is also possible to generate this interrupt when all the events associated with an ACM Timer are completed for an ACM trigger cycle. This interrupt can be cleared by writing to the relevant `W1C` (write 1 to clear) bit in `ACM_EVSTAT` register.

The Event Missed Interrupt is generated when an enabled event is missed for a trigger cycle and the corresponding mask bit of `ACM_MEVMSK` register is set. The event might be missed in Event comparator unit if more than one event, related to same Timer, are active during the same `SCLK1` cycle; or it can be missed in Timing Generation Unit if an event occur when event pending FIFO is full. The `ACM_MEVSTAT` register provides the status of each missed event indicating which event miss caused the interrupt. This interrupt can be cleared by writing the relevant `W1C` bit in the `ACM_MEVSTAT` register.

**NOTE:** A Status bit set either in `ACM_EVSTAT` or `ACM_MEVSTAT` registers triggers an interrupt only if the corresponding bit in the `ACM_EVMSK` or `ACM_MEVMSK` registers is enabled.

In addition to Event Completion interrupt (upon completion of particular event or all events), ACM can also provide an trigger output to Trigger Routing Unit of the processor. This trigger output can be used by Trigger slaves for their operations without requiring core intervention.

Event Order Registers

For debugging purpose, BF60x ACM hardware includes 16 Event Order registers-one per each event-which indicates the order in which the events were completed externally.

These registers are denoted as ACM\_EVORDx, where x stands for event ID, 0 to 15. The 8-bit EVT\_ORDER field of this register indicates in which order the ADC data has been captured corresponding to the event. This field accumulates the order count every trigger cycle, unless it is cleared in the software. At each trigger cycle, the values of the register will be updated, so it must be read at the end of the each trigger cycle (as it will write the new order value of the event in the next trigger cycle). Thus, the 8-bit field can store the order of 256 data captures at a stretch, after which it will start the order count from zero again.

All the event registers can be reset in software by setting ORDR\_RST bit of the ACM control register. When set, it clears all the Event Order Register value to zero. This bit auto-clears to 0 after all ACM\_EVORDx registers are cleared. These registers can also be cleared automatically by selected trigger input of ACM Timers, if AOREN bit is set. The OTSEL bit of the ACM Control register determines which trigger input to select for this auto-clearing.

The Event Order functionality is explained below with an example where ACM has only three events enabled (let it be Event1, Event7 and Event13). Following figure shows how the Order register value of these events will be at different stages.

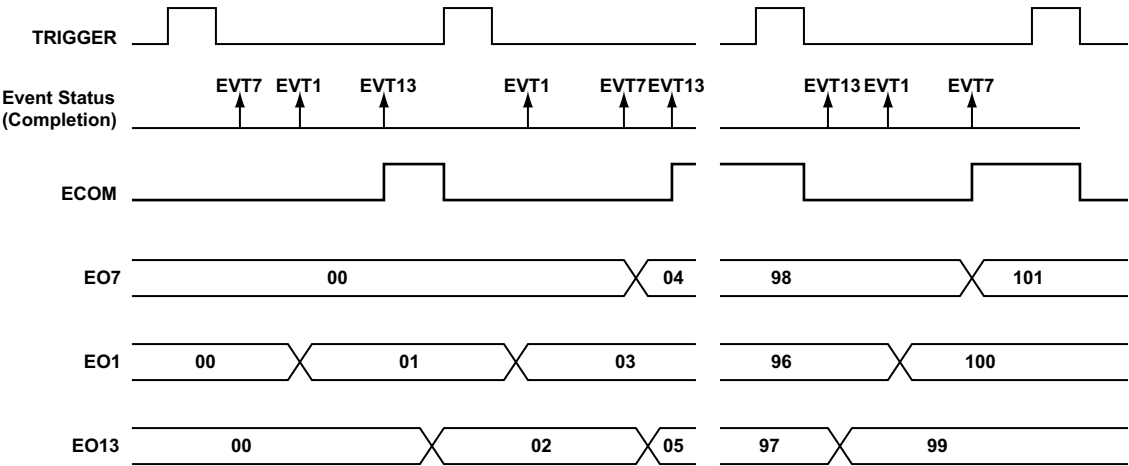


Figure 27-11:

ACM Register List

Table 27-1: ACM Module Registers

| Name    | Description          |
|---------|----------------------|
| ACM_CTL | ACM Control Register |

**Table 27-1:** ACM Module Registers (Continued)

| Name        | Description                                 |
|-------------|---------------------------------------------|
| ACM_TC0     | ACM Timing Configuration 0 Register         |
| ACM_TC1     | ACM Timing Configuration 1 Register         |
| ACM_STAT    | ACM Status Register                         |
| ACM_EVSTAT  | ACM Event Status Register                   |
| ACM_EVMSK   | ACM Completed Event Interrupt Mask Register |
| ACM_MEVSTAT | ACM Missed Event Status Register            |
| ACM_MEVMSK  | ACM Missed Event Interrupt Mask Register    |
| ACM_EVCTLn  | ACM Eventn Control Register                 |
| ACM_EVTIMEn | ACM Eventn Time Register                    |
| ACM_EVORDn  | ACM Eventn Order Register                   |
| ACM_TMR0    | ACM Timer 0 Register                        |
| ACM_TMR1    | ACM Timer 1 Register                        |

## ADSP-BF60x ACM Interrupt List

**Table 27-2:** ADSP-BF60x ACM Interrupt List

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| ACM0 Event Miss     | 38 |             | LEVEL       |
| ACM0 Event Complete | 39 |             | LEVEL       |

## ACM Programming Concepts

Since ACM module is being used with the SPORT, PWM, GPTIMER and GPIO; the programming must comply with the following guidelines for reliable operation of the ACM.

- ACM is control module and provides clock and chip select and control signals with required timing. But for capturing the data from ADC, one of the half of SPORT1 is used.
- The ACM should be enabled before enabling the SPORT. But SPORT can be configured before enabling ACM. SPORT should be configured in slave mode [external clock (ICLK=0), external frame sync(IFS=0)] as receiver.

The timings of external ADC decide the settings of LFS, LAFS, CKRE bits of SPORT Control register. Generally SPORT is configured in DSP serial mode to receive the ADC samples, but other operating mode (such as multichannel) may be possible.

If ACM is supposed to be programmed to gated clock mode (CLKMOD=1), the serial port should also be set in gated clock mode(GCLKEN=1).

- DMA mode of SPORT operation is preferred, as it saves the processor MIPS when receiving chunk of data. However, receiving ADC sample in core mode is also possible. So, when using DMA mode, DMA registers of selected SPORT should be configured appropriately; and DMA must be enabled before enabling SPORT. When using both primary and secondary channels of SPORT to receive data from two ADC channels, the 2D feature of DMA can be effectively used to de-interleave the data from two channels. When using Core mode of SPORT operation, core handler should be registered to handle the data read requests from SPORT receiver.
- In addition to SPORT register settings, the PORT registers should also be set properly to enable SPORT Data pins, ACM clock, CS and Control pins. When using either of the ACM\_T[1:0] trigger inputs for ACM Timers, FER and MUX bits of corresponding pins (PE8 or PG5) must be configured properly according to source of trigger input.
- When using ACM\_T[2:3] trigger inputs for ACM timers, the Trigger Routing Unit can be configured and enabled at this step. The corresponding Slave Trigger ID should be programmed properly using Slave Select Register to select the required master trigger. When using these trigger inputs, the Slave select register must not be configured when ACM is enabled, as default value of this register zero and Master Trigger ID-0 is system reserved.
- Before enabling the ACM (by setting the ACMEN bit), all the control bits of ACM Control register should be programmed properly. These control bits includes ACM trigger selects (TRGSELx), trigger input polarities (TRGPOLx), CS signal polarity (CSPOL), ACM clock polarity (CLKPOL), ACM clock mode (CLKMOD) and Serial port unit selection, Event order register settings.
- Configure the ACM Timing Control registers to define the ACM clock frequency and setup, hold & zero time of ACM control signals.
- The Timer Enabled (TMRENx) bits, however, should be programmed together only after ACM is enabled but once the bits are programmed it should not be changed later. Modifying these enable bits in the ACM Control register is not recommended while the ACM is in operation. Doing so can cause events



to change dependency from one timer to the other and can cause the values in the ACM status registers (ACM\_EVSTAT and ACM\_STAT) to be inaccurate. This means that if both timers are required for use, then enable them together after ACM is being enabled. If one timer is already enabled, then disable ACM, re-enable ACM and then program both timer enable bits together. Similarly when both timers are running, they should be disabled together.

- Once configuration peripherals are done, ACM should be enabled first and then SPORT module (SPORT DMA should be enabled before enabling SPORT, can be in previous steps also). Ideally trigger should not active when enabling the ACM.
- After enabling ACM, the event register pairs (Event Control and Event Time registers) should be configured and enabled to create required events.
- It should be noted that ACLK is an external clock relative to the SPORT peripheral. Therefore, any SPORT requirements around a minimum number of stable external clock cycles before assertion of the first SPORT frame sync need to be observed. The SPORT requires a minimum of 3 clock cycles before it is able to recognize a frame sync. When SPORT is configured in gated clock mode, this requirement becomes a minimum of 7 SPORT clock cycles. Therefore the required number of ACLK cycles should elapse before first assertion of CS. This can be guaranteed by any of the following methods:
  - Ensuring that ACM triggers are generated at least 3ACLK cycles after the ACM is enabled.
  - Ensuring that the event time value (ACM\_ETx) of the first active event is such that 3ACLK cycles would elapse before the event is processed.
- When the minimum number of ACLK cycles before the assertion of CS is not observed, the SPORT may miss the data of the first ADC sampling event. There can be a software workaround for fulfilling this requirement. Program the ACM\_TCO register (CKDIV value) after enabling the SPORT (subsequently after enabling ACM) but before the trigger is applied. Since the default value of CLKDIV is 1, the ACLK frequency is higher. Therefore the SPORT may receive the required clock cycles within short period of time (before the Frame Sync arrives).
- When using ACM\_T[2:3] trigger inputs, the master can be enabled at last (if it is configured only to provide triggers to ACM) to generate the triggers.
- While disabling the ACM system, the SPORT should be disabled first, then DMA and finally the ACM should be disabled.

## Emulation Mode Use Case

This section describes the usage modes of the ACM by illustrating how to implement various sequencing ADC sampling modes.

### Single-Shot Sequencing Mode Emulation

In single-shot sequencing mode, all enabled events are sequentially issued one after the other on the occurrence of an ACM trigger. The sequence of events is fixed, starting with Event0 and ending with Event15.

The following figure shows an example of single-shot sequencing mode where only Event0 and Event1 are enabled. ETIME0 is the value written into the ACM\_ET0 register, and ACMTMR0 is enabled in this mode. To emulate this mode of operation using the ACM.

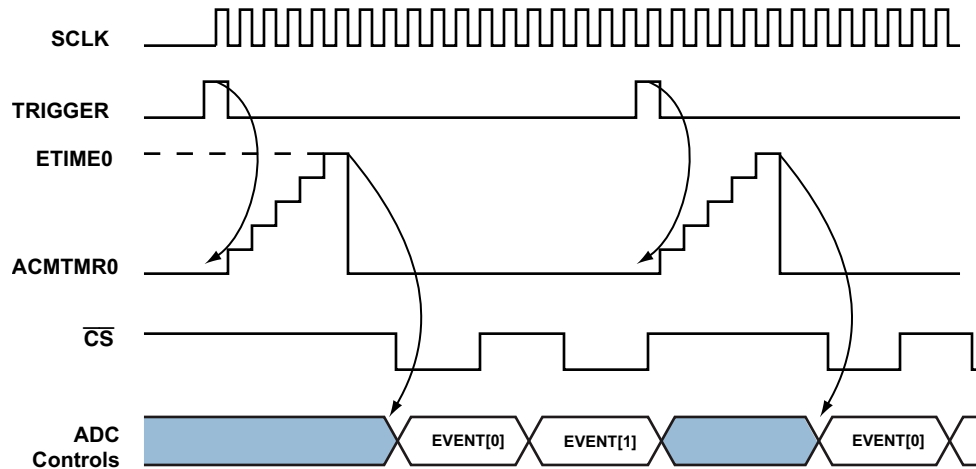


**Figure 27-12:** Single Shot Sequencing

- Configure the appropriate trigger source for initiating ACM activity. Please refer to “Interface Overview” for information on signals that can trigger the ACM counters.
- Enable only one ACM timer (ACMTMR0)
- Enable events and program the event time values as: Event0 time = X, Event1 time = X + Y, Event2 time = X + 2Y where:

$X = \text{ETIME0}$ , the initial time offset from trigger (if needed)

$Y = t_H + t_{CSW} + t_S + t_Z$ , where  $t_H$  is the hold time,  $t_Z$  is the zero t time, and  $t_S$  is the setup time for ACM Control lines as specified in ACM Timing Registers. For more information, refer "ACM External Pin Timing" section.

**NOTE:** Y has to be slightly less than the above value to ensure that the next event occurs before the first event completes, so that the next event is in the pending FIFO and enables the transitions between events without a break.

## Continuous Sequencing Mode Emulation

Continuous sequencing mode is similar to single-shot sequencing mode, except in continuous sequencing the event sequencing is continuously repeated. As in single-shot mode, the time offset is programmable in continuous mode. The trigger in continuous mode is relevant only for the first time. Therefore, any subsequent triggers after the first active edge of the trigger are neglected.

The following figure shows an example of continuous sequencing mode with only two events – Event0, Event1 enabled. To emulate continuous sequencing mode using ACM:

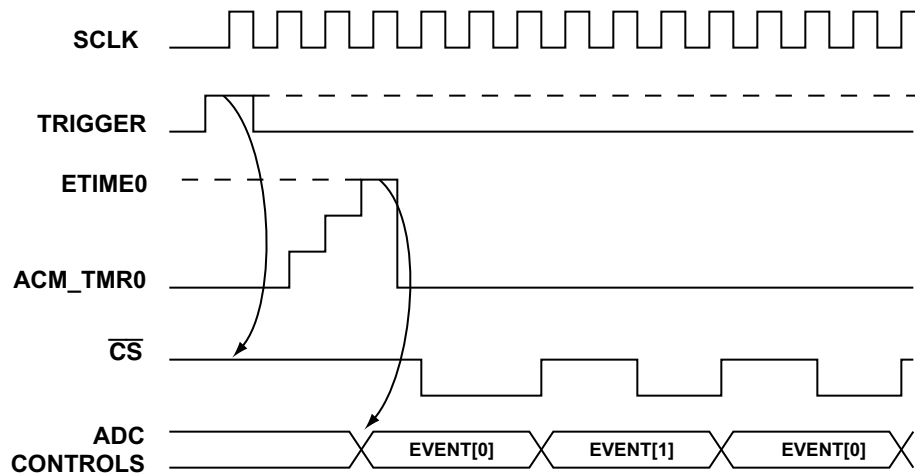


**Figure 27-13:** Continuous Sequencing

- Enable only one ACM Timer (say ACMTIMER0). Select Timer trigger input as from TRU of processor i.e. either of the ACM\_T[3:2].

If ACM\_T2 is selected, program ACM Slave Trigger ID-33 to select ACM Event Completion (whose master Trigger ID is 19) as trigger input. If ACM\_T3 is selected as trigger input, configure ACM Slave Trigger ID-34.

- Configure rest of the settings required by programming ACM Control and Timing Registers.
- Enable ACM Events with required ACM Control lines settings.

Program the Event time registers as: Event0 time = X, Event1 time = X + Y, Event 2 time = X + 2Y where:

X and Y values, in terms of SCLK1, are as described in the single-shot case (Y can be slightly less than  $t_H + t_{CSW} + t_S + t_Z$  to avoid any break between Events)

- Configure and enable System event controller.

Also configure, map and enable the ACM Event Completion interrupt which will interrupt upon completion of all enabled ACM Events for the current trigger. That means, set only ECOMP0 (or ECOMP1, if using ACMTIMER1) bit of ACM Event Interrupt Mask (ACM\_EVMSK) register.

- Enable ACM, SPORT and SPORT DMA as per the guidelines given in "Programming Concepts" section.
- Since we have configured ACM Trigger input as ACM Event Completion trigger output, the first trigger is necessary to start the ACM operation. We can provide this dummy trigger by writing Master Trigger ID into Master Trigger register (TRU0\_MTR).

So, write ACM Event Completion Trigger ID (19) into TRU0\_MTR register. This will trigger the ACM Timers and ACM will start handling the events.

- After completing all events, ACM will provide Event Completion Trigger and corresponding interrupt will be generated. The ACM Trigger output will be provided to ACM Timers which will reset it's counter and start running from zero, which will cause ACM to re-handle all the enabled ACM events. And this sequence continues.

However, in order to provide the Trigger outputs properly, the interrupt latch must be cleared in the ISR by clearing `ECOMP0S` (or `ECOMP1S`, if using `ACMTIMER1`) bit of the ACM Event Completion Status register (`ACM_EVSTAT`).

## ACM Register Descriptions

ACM (ACM) contains the following registers.

**Table 27-3:** ACM Module Registers

| Name                     | Description                                 |
|--------------------------|---------------------------------------------|
| <code>ACM_CTL</code>     | ACM Control Register                        |
| <code>ACM_TC0</code>     | ACM Timing Configuration 0 Register         |
| <code>ACM_TC1</code>     | ACM Timing Configuration 1 Register         |
| <code>ACM_STAT</code>    | ACM Status Register                         |
| <code>ACM_EVSTAT</code>  | ACM Event Status Register                   |
| <code>ACM_EVMSK</code>   | ACM Completed Event Interrupt Mask Register |
| <code>ACM_MEVSTAT</code> | ACM Missed Event Status Register            |
| <code>ACM_MEVMSK</code>  | ACM Missed Event Interrupt Mask Register    |
| <code>ACM_EVCTLn</code>  | ACM Eventn Control Register                 |
| <code>ACM_EVTIMEn</code> | ACM Eventn Time Register                    |
| <code>ACM_EVORDn</code>  | ACM Eventn Order Register                   |
| <code>ACM_TMR0</code>    | ACM Timer 0 Register                        |
| <code>ACM_TMR1</code>    | ACM Timer 1 Register                        |

## ACM Control Register

### ACM\_CTL: ACM Control Register - R/W

Reset = 0x0000 0000

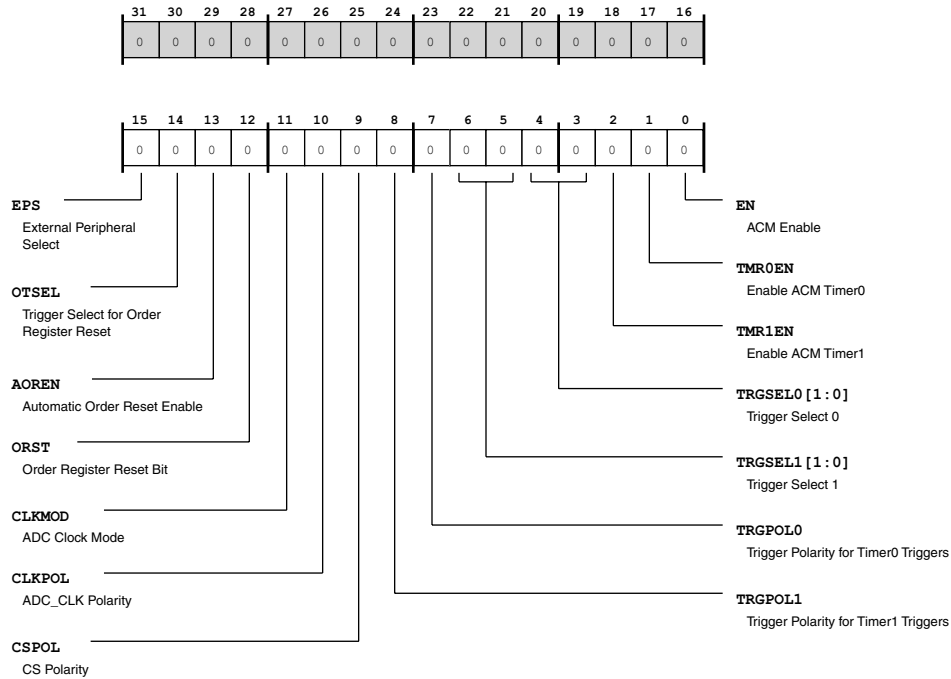


**Figure 27-14:** ACM\_CTL Register Diagram

**Table 27-4:** ACM\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                         |
|---------|----------|---------------------------------------------------------------------------------|
| 15      | EPS      | External Peripheral Select.                                                     |
|         |          | 0 SPORT2 interfaces to ACM                                                      |
|         |          | 1 SPORT1_B interfaces to ACM                                                    |
| 14      | OTSEL    | Trigger Select for Order Register Reset.<br>Applicable only if AOREN bit is set |
|         |          | 0 TMR0 trigger will reset the order registers                                   |
|         |          | 1 TMR1 trigger will reset the order registers                                   |

**Table 27-4:** ACM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                   |                                                        |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------|
| 13      | AOREN    | Automatic Order Reset Enable.<br>(OTSEL selects which Trigger)                                                                                            |                                                        |
|         |          | 0                                                                                                                                                         | Automatic Order Reset with Trigger is disabled         |
|         |          | 1                                                                                                                                                         | Automatic Order Reset with Trigger is enabled          |
| 12      | ORST     | Order Register Reset Bit.<br>If set it clears all the Event Order Register value to 0. This bit auto-clears to 0 after Event Order registers are cleared. |                                                        |
| 11      | CLKMOD   | ADC Clock Mode.                                                                                                                                           |                                                        |
|         |          | 0                                                                                                                                                         | ACM generates continuous ACLK                          |
|         |          | 1                                                                                                                                                         | ACLK is gated when CS is inactive                      |
| 10      | CLKPOL   | ADC_CLK Polarity.                                                                                                                                         |                                                        |
|         |          | 0                                                                                                                                                         | Falling edge of ADC_CLK comes after ADC_CS goes active |
|         |          | 1                                                                                                                                                         | Rising edge of ADC_CLK comes after ADC_CS goes active  |
| 9       | CSPOL    | CS Polarity.                                                                                                                                              |                                                        |
|         |          | 0                                                                                                                                                         | Active Low CS                                          |
|         |          | 1                                                                                                                                                         | Active High CS                                         |
| 8       | TRGPOL1  | Trigger Polarity for Timer1 Triggers.                                                                                                                     |                                                        |
|         |          | 0                                                                                                                                                         | Rising edge will create a trigger                      |
|         |          | 1                                                                                                                                                         | Falling edge will create a trigger                     |
| 7       | TRGPOL0  | Trigger Polarity for Timer0 Triggers.                                                                                                                     |                                                        |
|         |          | 0                                                                                                                                                         | Rising edge will create a trigger                      |
|         |          | 1                                                                                                                                                         | Falling edde will create a trigger                     |

**Table 27-4:** ACM\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                        |
|---------|----------|----------------------------------------------------------------|
| 6:5     | TRGSEL1  | Trigger Select 1.<br>Selects the external triggers for Timer 1 |
|         |          | 0 Select Trigger0 (ACM0_T0)                                    |
|         |          | 1 Select Trigger1 (ACM0_T1)                                    |
|         |          | 2 Select Trigger2 (Trigger Slave ID-33)                        |
|         |          | 3 Select Trigger3 (Trigger Slave ID-34)                        |
| 4:3     | TRGSEL0  | Trigger Select 0.<br>Selects the external triggers for Timer0  |
|         |          | 0 Select Trigger0 (ACM0_T0)                                    |
|         |          | 1 Select Trigger1 (ACM0_T1)                                    |
|         |          | 2 Select Trigger2 (Trigger Slave ID-33)                        |
|         |          | 3 Select Trigger3 (GPIOx/TMRx)                                 |
| 2       | TMR1EN   | Enable ACM Timer1.                                             |
|         |          | 0 Timer1 is disabled                                           |
|         |          | 1 Timer1 is enabled                                            |
| 1       | TMR0EN   | Enable ACM Timer0.                                             |
|         |          | 0 Timer0 is disabled                                           |
|         |          | 1 Timer0 is enabled                                            |
| 0       | EN       | ACM Enable.                                                    |
|         |          | 0 ACM is disabled                                              |
|         |          | 1 ACM is enabled                                               |

## ACM Timing Configuration 0 Register

ACM\_TC0: ACM Timing Configuration 0 Register - R/W

Reset = 0x0000 0001

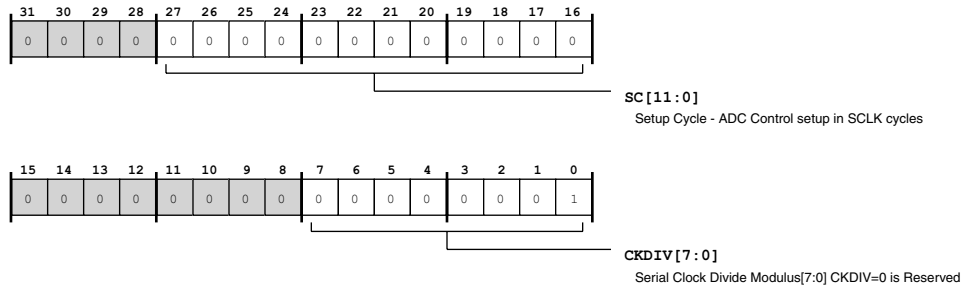


Figure 27-15: ACM\_TC0 Register Diagram

Table 27-5: ACM\_TC0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 27:16   | SC       | Setup Cycle - ADC Control setup in SCLK cycles.<br>Setup time for ADC Control signals (A[4:0]) in SCLK cycles with respect to active CS edge |
|         |          | 01 SCLK cycle setup time                                                                                                                     |
|         |          | 12 SCLK cycle setup time                                                                                                                     |
|         |          | 40954096 SCLK cycle setup time                                                                                                               |
| 7:0     | CKDIV    | Serial Clock Divide Modulus[7:0] CKDIV=0 is Reserved.                                                                                        |

ACM Timing Configuration 1 Register



**ACM\_TC1: ACM Timing Configuration 1 Register - R/W**

Reset = 0x0000 000f

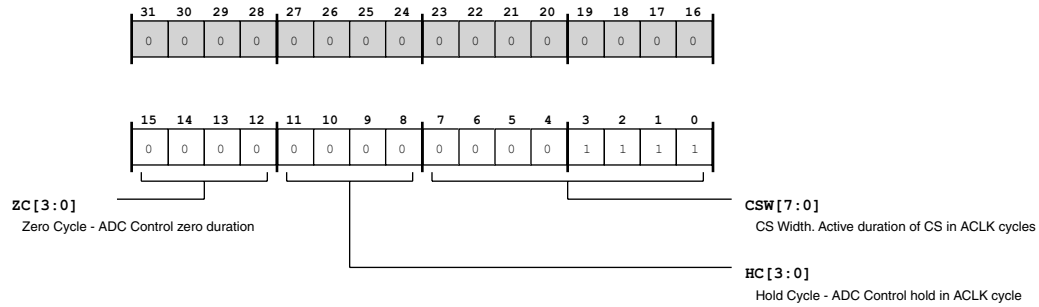


**Figure 27-16:** ACM\_TC1 Register Diagram

**Table 27-6:** ACM\_TC1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 15:12   | ZC       | Zero Cycle - ADC Control zero duration.<br>All ADC controls will be driven low for ZC ACLK cycles                              |
|         |          | 0 0 ACLK cycle ADC Controls are zero                                                                                           |
|         |          | 1 1 ACLK cycle ADC controls are zero                                                                                           |
|         |          | 15 15 ACLK cycle ADC controls are zero                                                                                         |
| 11:8    | HC       | Hold Cycle - ADC Control hold in ACLK cycle.<br>All ADC controls will be held after the inactive edge of CS for HC ACLK cycles |
|         |          | 0 0 ACLK hold cycle                                                                                                            |
|         |          | 1 1 ACLK hold cycle                                                                                                            |
|         |          | 15 15 ACLK hold cycle                                                                                                          |
| 7:0     | CSW      | CS Width. Active duration of CS in ACLK cycles.                                                                                |
|         |          | 0 1 ACLK active CS                                                                                                             |
|         |          | 1 2 ACLK active CS                                                                                                             |
|         |          | 15 16 ACLK active CS                                                                                                           |
|         |          | 255 256 ACLK active CS                                                                                                         |

ACM Status Register

ACM\_STAT: ACM Status Register - R/NW

Reset = 0x0000 0000

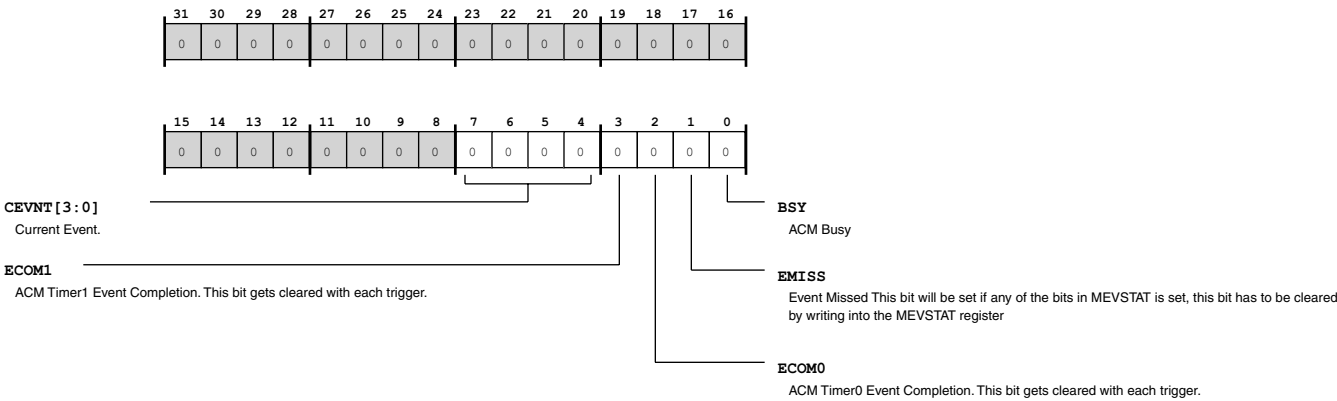


Figure 27-17: ACM\_STAT Register Diagram

Table 27-7: ACM\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                |                                                                 |
|---------|----------|------------------------------------------------------------------------|-----------------------------------------------------------------|
| 7:4     | CEVNT    | Current Event..                                                        |                                                                 |
|         |          | 0                                                                      | Current Event corresponds to Event0 Register                    |
|         |          | 1                                                                      | Current Event corresponds to Event1 Register                    |
|         |          | 15                                                                     | Current Event corresponds to Event15 Register                   |
| 3       | ECOM1    | ACM Timer1 Event Completion. This bit gets cleared with each trigger.. |                                                                 |
|         |          | 0                                                                      | Timer1 Events are not completed for the current trigger         |
|         |          | 1                                                                      | All enabled Timer1 events are completed for the current trigger |

**Table 27-7:** ACM\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                  |                                                                             |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| 2       | ECOM0    | ACM Timer0 Event Completion. This bit gets cleared with each trigger..                                                                   |                                                                             |
|         |          | 0                                                                                                                                        | Timer0 Events are not completed for the current trigger                     |
|         |          | 1                                                                                                                                        | All enabled Timer0 events are completed for the current trigger             |
| 1       | EMISS    | Event Missed This bit will be set if any of the bits in MEVSTAT is set, this bit has to be cleared by writing into the MEVSTAT register. |                                                                             |
|         |          | 0                                                                                                                                        | Events not missed                                                           |
|         |          | 1                                                                                                                                        | Events were missed                                                          |
| 0       | BSY      | ACM Busy.                                                                                                                                |                                                                             |
|         |          | 0                                                                                                                                        | ACM is idle                                                                 |
|         |          | 1                                                                                                                                        | An external sampling event in progress (CS is active or about to go active) |

## ACM Event Status Register

ACM\_EVSTAT: ACM Event Status Register - R/W

Reset = 0x0000 0000

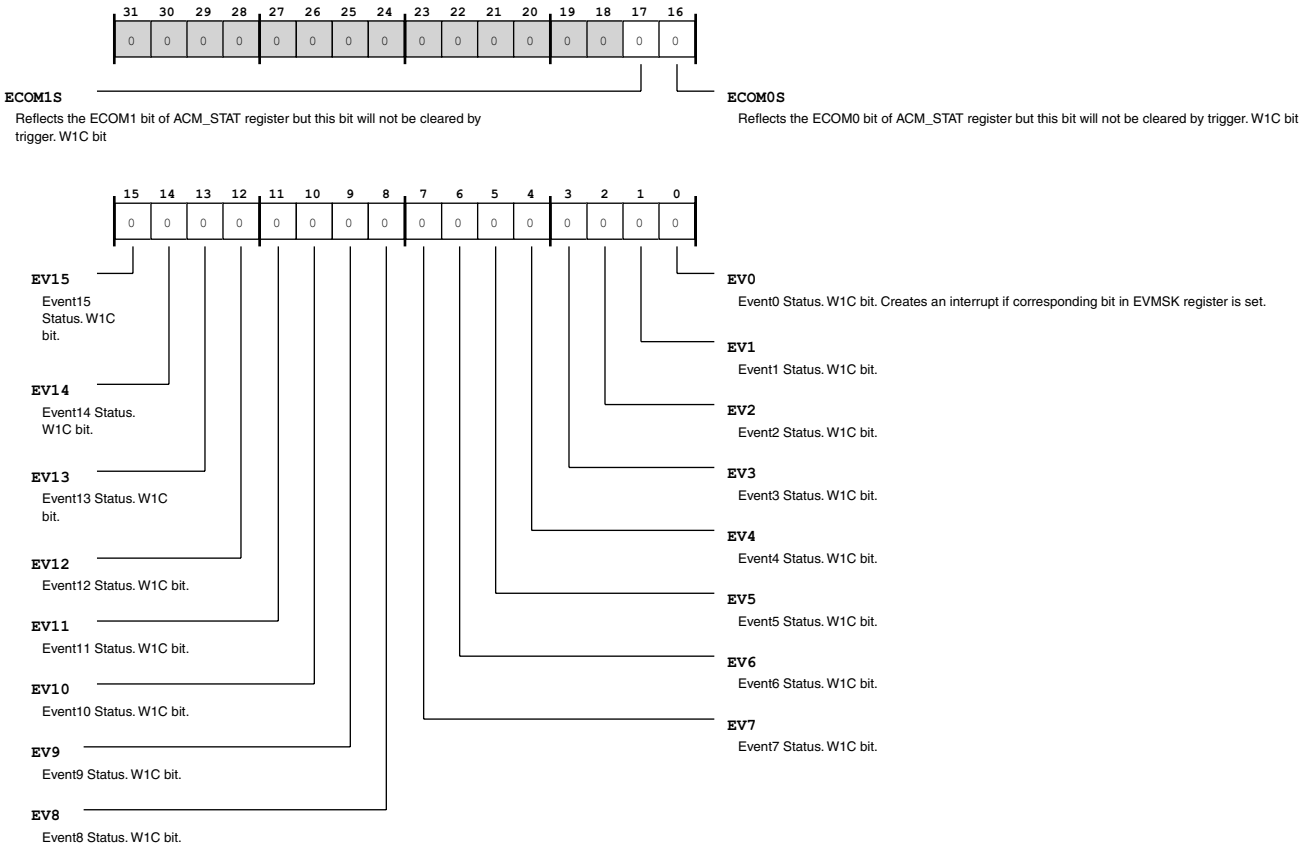


Figure 27-18: ACM\_EVSTAT Register Diagram

Table 27-8: ACM\_EVSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                           |
|---------|----------|---------------------------------------------------------------------------------------------------|
| 17      | ECOM1S   | Reflects the ECOM1 bit of ACM_STAT register but this bit will not be cleared by trigger. W1C bit. |
| 16      | ECOM0S   | Reflects the ECOM0 bit of ACM_STAT register but this bit will not be cleared by trigger. W1C bit. |
| 15      | EV15     | Event15 Status. W1C bit..                                                                         |
| 14      | EV14     | Event14 Status. W1C bit..                                                                         |
| 13      | EV13     | Event13 Status. W1C bit..                                                                         |
| 12      | EV12     | Event12 Status. W1C bit..                                                                         |

**Table 27-8:** ACM\_EVSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------|
| 11      | EV11     | Event11 Status. W1C bit..                                                                    |
| 10      | EV10     | Event10 Status. W1C bit..                                                                    |
| 9       | EV9      | Event9 Status. W1C bit..                                                                     |
| 8       | EV8      | Event8 Status. W1C bit..                                                                     |
| 7       | EV7      | Event7 Status. W1C bit..                                                                     |
| 6       | EV6      | Event6 Status. W1C bit..                                                                     |
| 5       | EV5      | Event5 Status. W1C bit..                                                                     |
| 4       | EV4      | Event4 Status. W1C bit..                                                                     |
| 3       | EV3      | Event3 Status. W1C bit..                                                                     |
| 2       | EV2      | Event2 Status. W1C bit..                                                                     |
| 1       | EV1      | Event1 Status. W1C bit..                                                                     |
| 0       | EV0      | Event0 Status. W1C bit. Creates an interrupt if corresponding bit in EVMSK register is set.. |
|         |          | 0 ACM has not begun/Completed conversion for Event0                                          |
|         |          | 1 ACM has Completed the conversion for Event0.                                               |

## ACM Completed Event Interrupt Mask Register

ACM\_EVMSK: ACM Completed Event Interrupt Mask Register - R/W

Reset = 0x0000 0000

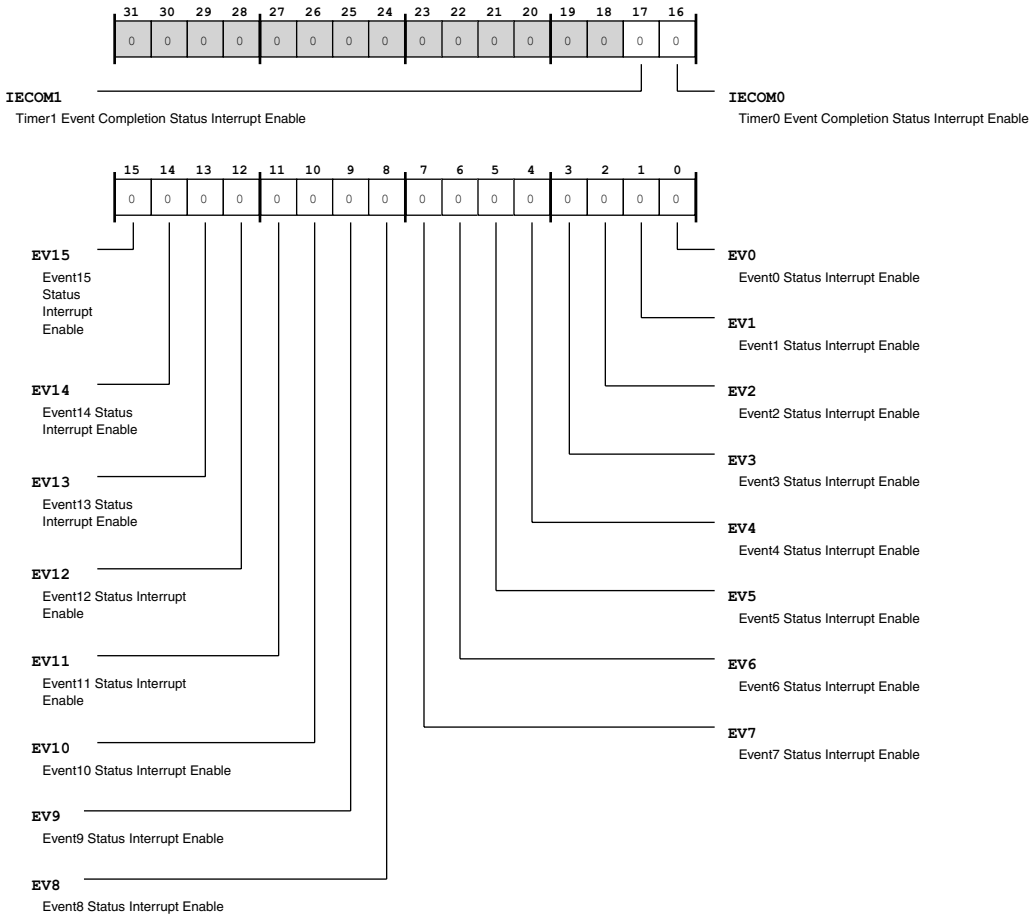


Figure 27-19: ACM\_EVMSK Register Diagram

Table 27-9: ACM\_EVMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                 |
|---------|----------|-------------------------------------------------------------------------|
| 17      | IECOM1   | Timer1 Event Completion Status Interrupt Enable.                        |
| 16      | IECOM0   | Timer0 Event Completion Status Interrupt Enable.                        |
|         | 0        | Interrupts for ECOM0S is disabled                                       |
|         | 1        | Interrupt will be generated when ECOM0S bit is set in ACM_STAT register |
| 15      | EV15     | Event15 Status Interrupt Enable.                                        |

**Table 27-9:** ACM\_EVMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------|
| 14      | EV14     | Event14 Status Interrupt Enable.                                                                                   |
| 13      | EV13     | Event13 Status Interrupt Enable.                                                                                   |
| 12      | EV12     | Event12 Status Interrupt Enable.                                                                                   |
| 11      | EV11     | Event11 Status Interrupt Enable.                                                                                   |
| 10      | EV10     | Event10 Status Interrupt Enable.                                                                                   |
| 9       | EV9      | Event9 Status Interrupt Enable.                                                                                    |
| 8       | EV8      | Event8 Status Interrupt Enable.                                                                                    |
| 7       | EV7      | Event7 Status Interrupt Enable.                                                                                    |
| 6       | EV6      | Event6 Status Interrupt Enable.                                                                                    |
| 5       | EV5      | Event5 Status Interrupt Enable.                                                                                    |
| 4       | EV4      | Event4 Status Interrupt Enable.                                                                                    |
| 3       | EV3      | Event3 Status Interrupt Enable.                                                                                    |
| 2       | EV2      | Event2 Status Interrupt Enable.                                                                                    |
| 1       | EV1      | Event1 Status Interrupt Enable.                                                                                    |
| 0       | EV0      | Event0 Status Interrupt Enable.                                                                                    |
|         |          | 0 Interrupts for Event0 is disabled                                                                                |
|         |          | 1 An interrupt will be generated whenever Event0 Status(EV0 in EVSTAT) gets set i.e interrupt on Event0 completion |

## ACM Missed Event Status Register

ACM\_MEVSTAT: ACM Missed Event Status Register - R/W

Reset = 0x0000 0000

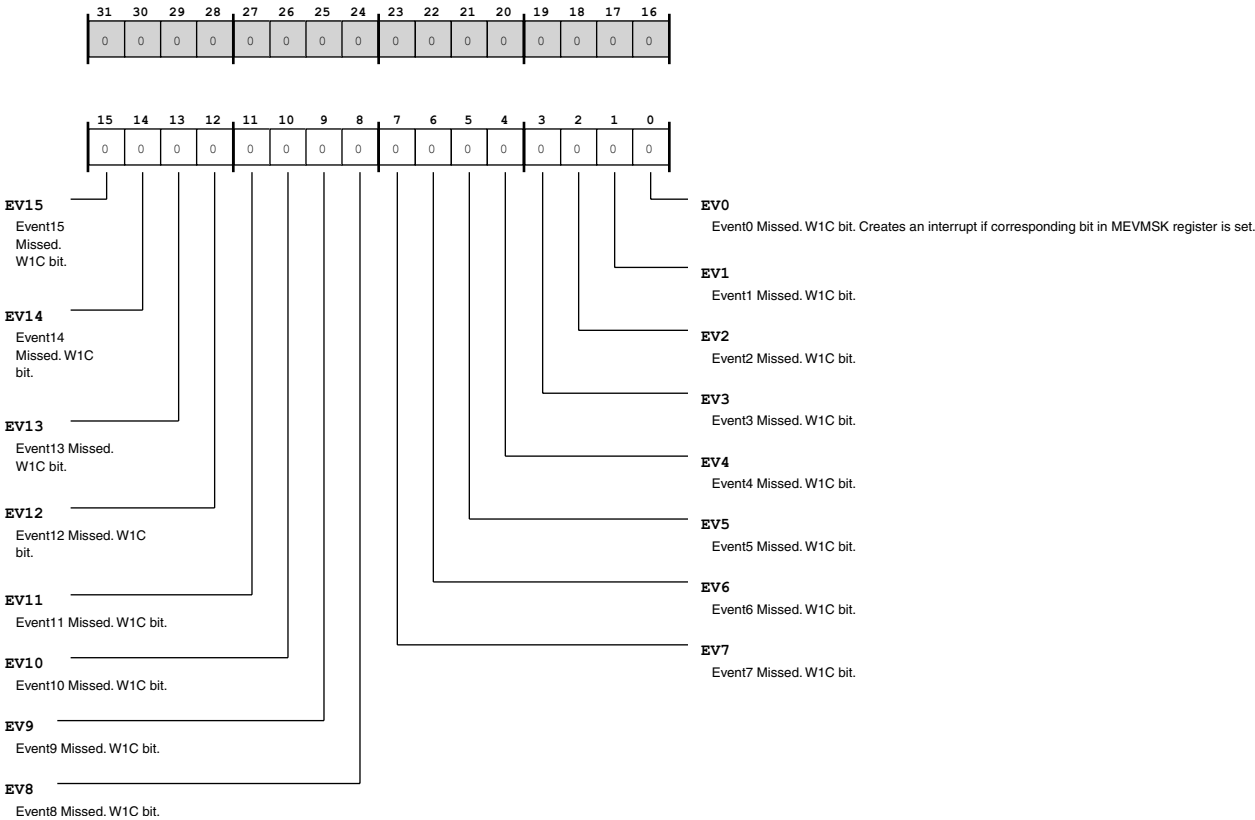


Figure 27-20: ACM\_MEVSTAT Register Diagram

Table 27-10: ACM\_MEVSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 15      | EV15     | Event15 Missed. W1C bit.. |
| 14      | EV14     | Event14 Missed. W1C bit.. |
| 13      | EV13     | Event13 Missed. W1C bit.. |
| 12      | EV12     | Event12 Missed. W1C bit.. |
| 11      | EV11     | Event11 Missed. W1C bit.. |
| 10      | EV10     | Event10 Missed. W1C bit.. |
| 9       | EV9      | Event9 Missed. W1C bit..  |



**Table 27-10:** ACM\_MEVSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------|
| 8       | EV8      | Event8 Missed. W1C bit..                                                                      |
| 7       | EV7      | Event7 Missed. W1C bit..                                                                      |
| 6       | EV6      | Event6 Missed. W1C bit..                                                                      |
| 5       | EV5      | Event5 Missed. W1C bit..                                                                      |
| 4       | EV4      | Event4 Missed. W1C bit..                                                                      |
| 3       | EV3      | Event3 Missed. W1C bit..                                                                      |
| 2       | EV2      | Event2 Missed. W1C bit..                                                                      |
| 1       | EV1      | Event1 Missed. W1C bit..                                                                      |
| 0       | EV0      | Event0 Missed. W1C bit. Creates an interrupt if corresponding bit in MEVMSK register is set.. |
|         |          | 0      Event0 has not been missed since the last trigger                                      |
|         |          | 1      Event0 was missed.                                                                     |

## ACM Missed Event Interrupt Mask Register

ACM\_MEVMSK: ACM Missed Event Interrupt Mask Register - R/W

Reset = 0x0000 0000

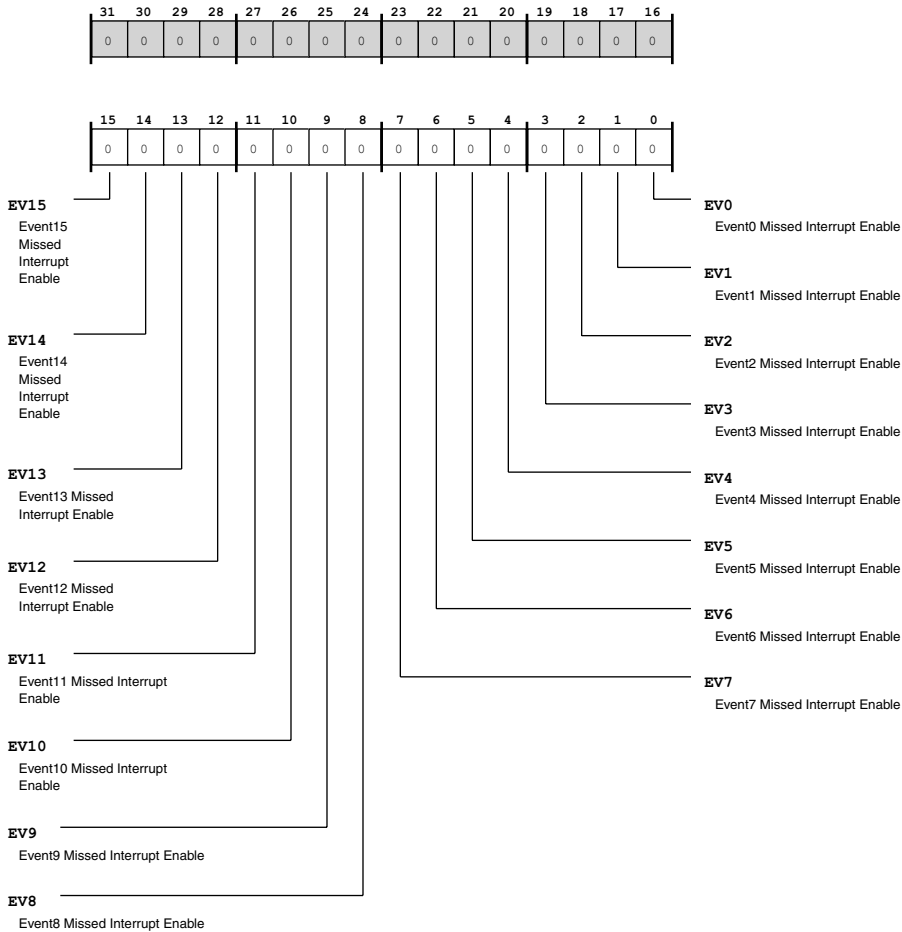


Figure 27-21: ACM\_MEVMSK Register Diagram

Table 27-11: ACM\_MEVMSK Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 15      | EV15     | Event15 Missed Interrupt Enable. |
| 14      | EV14     | Event14 Missed Interrupt Enable. |
| 13      | EV13     | Event13 Missed Interrupt Enable. |
| 12      | EV12     | Event12 Missed Interrupt Enable. |
| 11      | EV11     | Event11 Missed Interrupt Enable. |

**Table 27-11:** ACM\_MEVMSK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration          |                                                                                                                |
|---------|----------|----------------------------------|----------------------------------------------------------------------------------------------------------------|
| 10      | EV10     | Event10 Missed Interrupt Enable. |                                                                                                                |
| 9       | EV9      | Event9 Missed Interrupt Enable.  |                                                                                                                |
| 8       | EV8      | Event8 Missed Interrupt Enable.  |                                                                                                                |
| 7       | EV7      | Event7 Missed Interrupt Enable.  |                                                                                                                |
| 6       | EV6      | Event6 Missed Interrupt Enable.  |                                                                                                                |
| 5       | EV5      | Event5 Missed Interrupt Enable.  |                                                                                                                |
| 4       | EV4      | Event4 Missed Interrupt Enable.  |                                                                                                                |
| 3       | EV3      | Event3 Missed Interrupt Enable.  |                                                                                                                |
| 2       | EV2      | Event2 Missed Interrupt Enable.  |                                                                                                                |
| 1       | EV1      | Event1 Missed Interrupt Enable.  |                                                                                                                |
| 0       | EV0      | Event0 Missed Interrupt Enable.  |                                                                                                                |
|         |          | 0                                | Interrupts for Event0 Miss is disabled                                                                         |
|         |          | 1                                | An interrupt will be generated whenever Event0 is Missed(EV0 in MEVSTAT) gets set i.e interrupt on Event0 Miss |

## ACM Eventn Control Register

ACM\_EVCTLn: ACM Eventn Control Register - R/W

Reset = 0x0000 0000

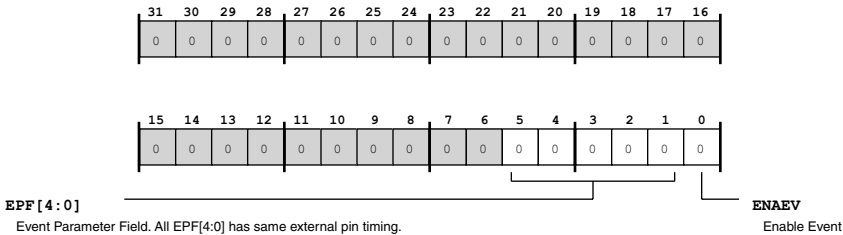


Figure 27-22: ACM\_EVCTLn Register Diagram

Table 27-12: ACM\_EVCTLn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                      |                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| 5:1     | EPF      | Event Parameter Field. All EPF[4:0] has same external pin timing.. The Event Parameter Field can be flexibly mapped to the various ADC controls based on the type of ADC, usage mode. For more information, see the operating modes section. |                                                                                                                                      |
| 0       | ENAEV    | Enable Event.                                                                                                                                                                                                                                |                                                                                                                                      |
|         |          | 0                                                                                                                                                                                                                                            | Event Disabled, Corresponding ACM_ETx register has no significance, the control values will not be used                              |
|         |          | 1                                                                                                                                                                                                                                            | Event Enabled, a matching time or a external trigger can cause a sampling event to happen to the ADC with the specified ADC controls |

ACM Eventn Time Register

**ACM\_EVTIMEn: ACM Eventn Time Register - R/W**

Reset = 0x0000 0000

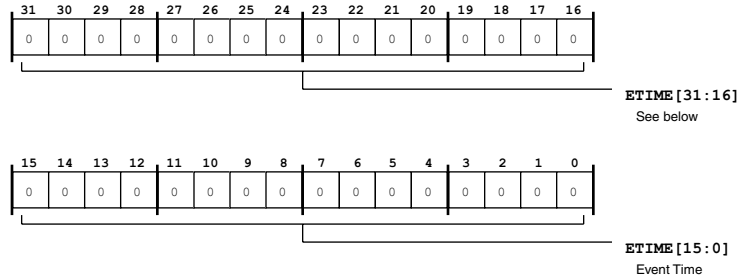


**Figure 27-23:** ACM\_EVTIMEn Register Diagram

**Table 27-13:** ACM\_EVTIMEn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | ETIME    | Event Time.             |

## ACM Eventn Order Register

**ACM\_EVORDn: ACM Eventn Order Register - R/NW**

Reset = 0x0000 0000

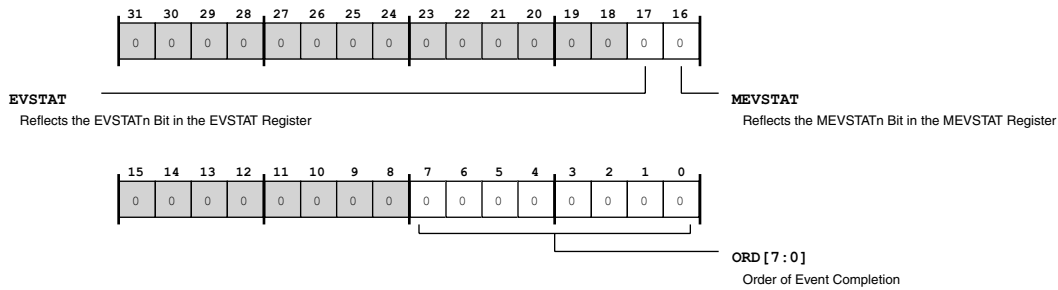


**Figure 27-24:** ACM\_EVORDn Register Diagram

**Table 27-14:** ACM\_EVORDn Register Fields

| Bit No. | Bit Name | Description/Enumeration                            |
|---------|----------|----------------------------------------------------|
| 17      | EVSTAT   | Reflects the EVSTATn Bit in the EVSTAT Register.   |
| 16      | MEVSTAT  | Reflects the MEVSTATn Bit in the MEVSTAT Register. |

Table 27-14: ACM\_EVORDn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration    |                                                                                                       |
|---------|----------|----------------------------|-------------------------------------------------------------------------------------------------------|
| 7:0     | ORD      | Order of Event Completion. |                                                                                                       |
|         |          | 0                          | Eventx was the first Event to be completed once the ACM was enabled or after the ORDR_RST bit was set |
|         |          | 1                          | Eventx was the second Event to be completed                                                           |
|         |          | 255                        | Eventx was the 256th Event to be completed after ACM enable or Order Reset                            |

ACM Timer 0 Register

ACM\_TMR0: ACM Timer 0 Register - R/W

Reset = 0x0000 0000

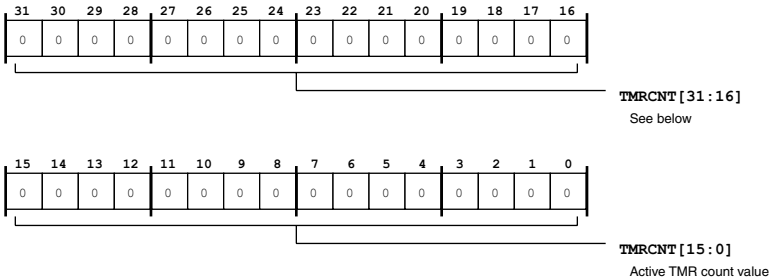


Figure 27-25: ACM\_TMR0 Register Diagram

Table 27-15: ACM\_TMR0 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TMRCNT   | Active TMR count value.<br>These bits can be written also, but this feature is given only for testability. The write feature should not be documented for user. |

## ACM Timer 1 Register

ACM\_TMR1: ACM Timer 1 Register - R/W

Reset = 0x0000 0000

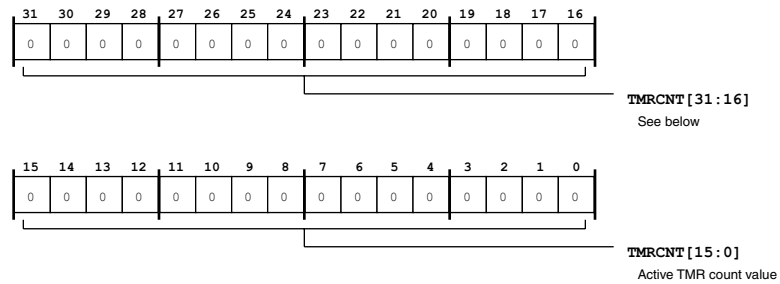


**Figure 27-26:** ACM\_TMR1 Register Diagram

**Table 27-16:** ACM\_TMR1 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | TMRCNT   | Active TMR count value.<br>These bits can be written also, but this feature is given only for testability. The write feature should not be documented for user. |





## 28 Link Port (LP)

Link ports allow the processor to connect to other processors or peripheral link ports using a simple communication protocol for high-speed parallel data transfer. This peripheral allows a variety of I/O peripheral interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes.

The ADSP-BF60x processor has four link ports and each supports 8-bit wide data transfers. The link port pins are multiplexed in the GPIO ports as follows.

- Link port 0 and 1 signals are multiplexed in GPIO ports A and B.
- Link port 2 and 3 signals are multiplexed in GPIO ports E and F.

Each of these identical modules can operate independently and simultaneously, allowing glueless high-speed connectivity of up to four external processors in BF60x.

### LP Features

All link ports are identical in their design and have the following common features.

- Bidirectional ports with eight data lines, an acknowledge line, and a clock line.
- Provide high-speed, point-to-point data transfers to other processors, allowing different types of interconnections between multiple DSPs
- Pack data into 32-bit words. This data can be directly read by the processor or transferred via DMA to or from on-chip memory.
- Support for data buffering through 2-deep FIFO for transmit and 4-deep FIFO for receive.
- Programmable clock and acknowledge based handshake mechanism for efficient communication.
- Dedicated DMA channel.

### LP Functional Description

This section provides a description of the link port, including a list of its registers and a functional block diagram.

## LP Register List

The LP are 8-bit wide ports, which can connect to another processor or peripheral LP. These ports allow a variety of interconnection schemes to I/O peripheral devices as well as co-processing and multiprocessing schemes. A set of registers govern LP operations. For more information on LP functionality, see the LP register descriptions.

**Table 28-1:** LP Module Registers

| Name          | Description                   |
|---------------|-------------------------------|
| LP_CTL        | Control Register              |
| LP_STAT       | Status Register               |
| LP_DIV        | Clock Divider Value           |
| LP_TX         | Transmit Buffer               |
| LP_RX         | Receive Buffer                |
| LP_TXIN_SHDW  | Shadow Input Transmit Buffer  |
| LP_TXOUT_SHDW | Shadow Output Transmit Buffer |

## ADSP-BF60x LP Interrupt List

**Table 28-2:** ADSP-BF60x LP Interrupt List

| Description     | ID | DMA Channel | Sensitivity |
|-----------------|----|-------------|-------------|
| LP0 DMA Channel | 72 | 13          | LEVEL       |
| LP0 Status      | 73 |             | LEVEL       |
| LP1 DMA Channel | 74 | 14          | LEVEL       |
| LP1 Status      | 75 |             | LEVEL       |
| LP2 DMA Channel | 76 | 15          | LEVEL       |
| LP2 Status      | 77 |             | LEVEL       |
| LP3 DMA Channel | 78 | 16          | LEVEL       |

**Table 28-2:** ADSP-BF60x LP Interrupt List (Continued)

| Description | ID | DMA Channel | Sensitivity |
|-------------|----|-------------|-------------|
| LP3 Status  | 79 |             | LEVEL       |

## ADSP-BF60x LP Trigger List

**Table 28-3:** Masters ADSP-BF60x LP Trigger List

| Description     | ID | Sensitivity |
|-----------------|----|-------------|
| LP0 DMA Channel | 35 | PULSE/EDGE  |
| LP1 DMA Channel | 36 | PULSE/EDGE  |
| LP2 DMA Channel | 37 | PULSE/EDGE  |
| LP3 DMA Channel | 38 | PULSE/EDGE  |
|                 |    |             |

**Table 28-4:** Slaves ADSP-BF60x LP Trigger List

| Description     | ID | Sensitivity |
|-----------------|----|-------------|
| LP0 DMA Channel | 35 |             |
| LP1 DMA Channel | 36 |             |
| LP2 DMA Channel | 37 |             |
| LP3 DMA Channel | 38 |             |
|                 |    |             |

## LP Block Diagram

The block diagram of a link port is shown in the following figure.

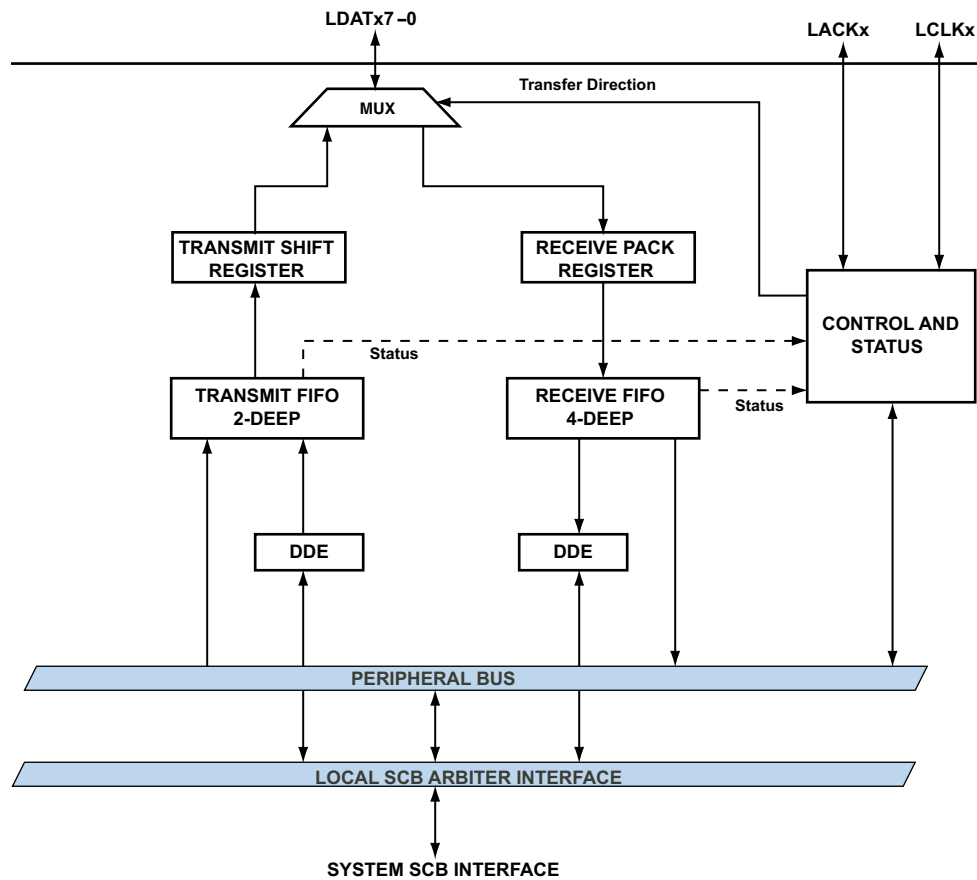
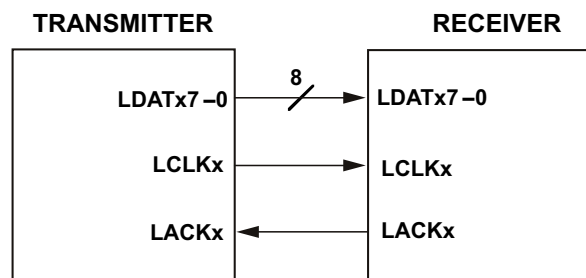


Figure 28-1: Link Port Block Diagram

## External Connections

A link port has eight data lines (LDAT7-0), a clock line (LCLK), and an acknowledge line (LACK). A link port can act as either (one at a time) a transmitter or a receiver.



X DENOTES THE LINK PORT NUMBER, 0-4

Figure 28-2: Link Port Pin Connections

External pull downs should be used for the LCLK and LACK pins so that the transmitter and receiver can be enabled, irrespective of the state of the other.

## Internal Blocks

As shown in the [Link Port Pin Connections](#) figure, the link ports have independent modules for transmit and receive. If enabled as transmitter, the link port uses a 2 deep 32-bit FIFO. If enabled as a receiver, the port uses a 4 deep 32-bit FIFO. These FIFOs can be accessed by the core MMR access bus as well as through the distributed DMA engines (DDE) through the system cross bar (SCB) interface. The `LTRAN` bit in the link port control register determines whether the module is enabled for transmit or receive operation.

## LP Architectural Concepts

This section describes link port architectural concepts, including link port protocol, buffers, intercommunication, clocking, and multi-processor connectivity.

### Link Port Protocol

A link port transmitted word consists of 4 bytes. The transmitter asserts the Link port clock (`LCLK`) with each byte of data. The falling edge of `LCLK` driven by the transmitter is used by the receiver to latch the byte. When the receiver is ready to accept another word in the receive buffer it asserts the acknowledge signal, `LACK`. The transmitter samples `LACK` driven by the receiver at the beginning of each word transmission. If `LACK` is de-asserted at that time, the transmitter does not transmit the next word. The transmitter leaves `LCLK` high and continues to drive the first byte of the next word until `LACK` is asserted. When this assertion occurs, `LCLK` is driven low by the transmitter and the transmission of the next word starts. If the transmit buffer is empty, `LCLK` remains low until the buffer is refilled, regardless of the state of `LACK`.

The `LACK` signal may de-assert when it anticipates that the buffer may fill. The `LACK` signal is reasserted by the receiver as soon as the internal DMA grant signal has occurred or the core reads the receive buffer. Either of these actions frees a buffer location.

**NOTE:** The `LACK` signal inhibits transmission of the next word and not of the current byte.

Link port acknowledge (`LACK`) provides a handshake between the receiver and transmitter in the following configurations.

- When configured as a transmitter, the port drives both the data and the clock while `LACK` is three-stated. In this mode `LCLK` is always synchronous with `SCLK`.
- When configured as a receiver, the link port drives the acknowledge signal and the data and clock lines are three-stated. In this case, external `LCLK` can either be synchronous or asynchronous with `SCLK`.
- When the link port is disabled, data, clock and acknowledge signals are three-stated.

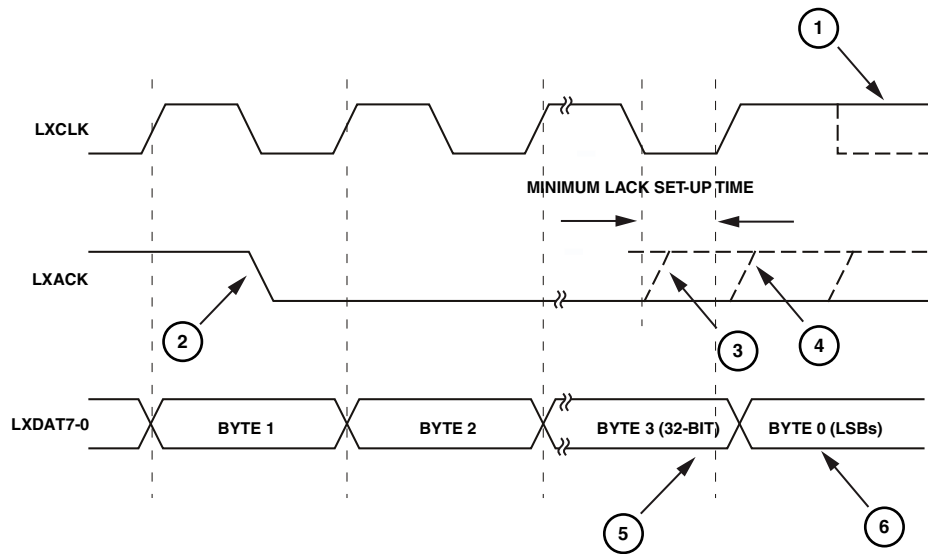


**Figure 28-3:** Link Port Communication and Handshake Waveform

The following list describes the stages shown in the *Link Port Communication and Handshake Waveform* figure.

1. LCLK stays high at byte 0 if LACK is sampled low on the previous LCLK rising edge. LCLK high indicates a stall.
2. The LACK signal may de-assert after byte 0.
3. The LACK signal reasserts as soon as the link buffer is not full (depending on RX FIFO conditions).
4. The transmitter samples LACK to determine whether to transmit the next word.
5. The receiver accepts the remaining word even if LACK is de-asserted. The transmitter does not send the following word.
6. Transmission of data for next word is held until LACK is asserted.

The acknowledge signal (LACK) is sampled by the transmitter and if it is high, the transmitter gives out the falling edges of LCLK for data sampling. The LACK signal is first sampled at the rising edge of SCLK, is further synchronized by one more SCLK stage and then this synchronized signal is given to the subsequent logic. The LCLK falling edge is aligned with SCLK falling edge in a 1:1 clock ratio mode and with the SCLK rising edge for the rest of the clock ratios. The following figures explain how the synchronization is maintained between the LACK and LCLK signals.

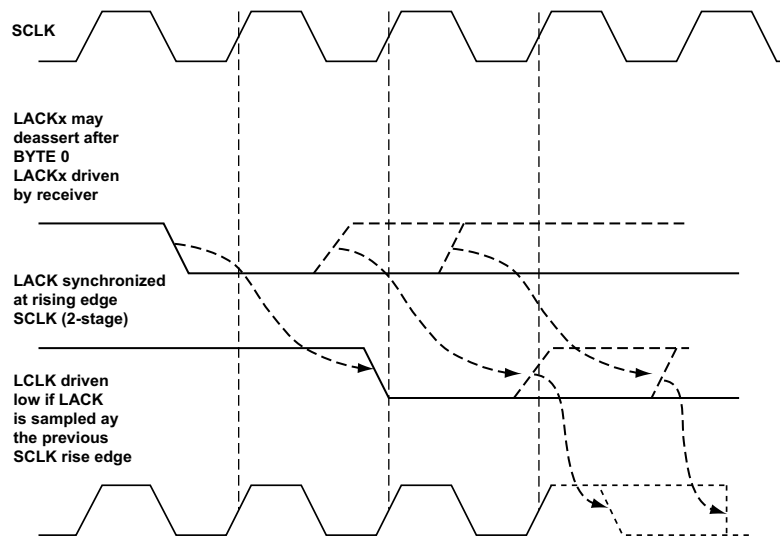


**Figure 28-4:** LACK synchronization for SCLK:LCLK=1:1 (synchronizing time is guaranteed to be 1.5 SCLK cycles).

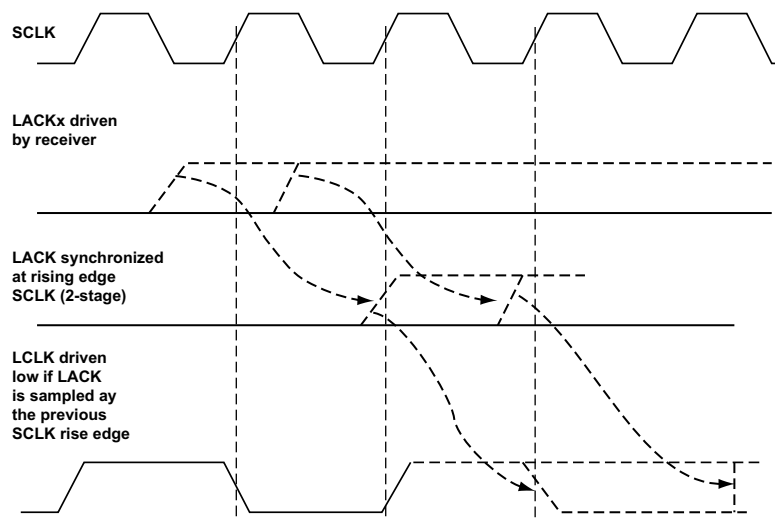


**Figure 28-5:** LACK synchronization for SCLK: LCLK=1:2, 1:4 and Up (synchronizing time is guaranteed to be 2 SCLK cycles).

The frequency of the link port clock (LCLK) is determined by the value programmed in the CLKDIV register at the transmitter. However, the signal appearing on the LCLK pin is also dependent on the status of the LACK pin driven by the receiver. The below figure explains this relationship.

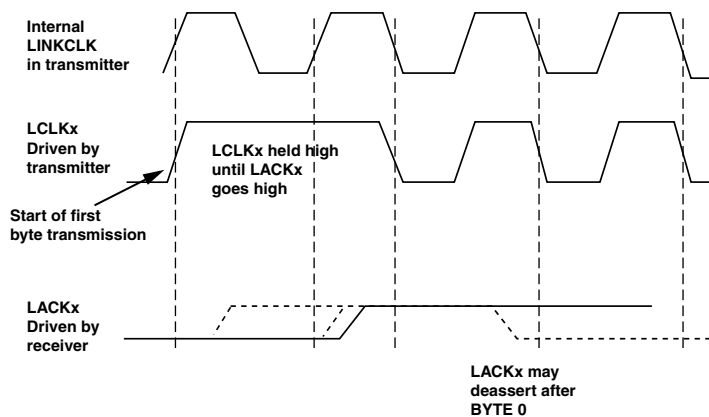


**Figure 28-6:** Relationship Between Internal Link Port Clock and Link Port Clock at the Pins

## Link Port Buffers

When a link port is configured as transmitter, a 2-deep FIFO buffer is used. A shift register unpacks the single 32-bit word to four 8-bit data bytes. As the FIFO gets space for more data, a new DMA request is made. The core can access FIFO through the LP\_TX register. If the FIFO becomes empty, the LCLK signal is de-asserted. Three writes (2 stage FIFO and 1 Shift Register) can occur to the transmit buffer by the core or DMA before it signals a full condition. The status of the FIFO is reflected by the FFST field in the LPSTAT register. Shift Register full/empty condition is not provided to the user. However, the user may poll the LPBS bit in the LPSTAT to know if link port is driving data from Shift Register to the pins or not. LPBS is also set when receiver has held off transmission by driving LACK low.

**NOTE:** When the 2-deep FIFO and the output shift register overflows, any further write in to the link port buffer shall overwrite the input stage of the FIFO.

Transmit FIFO can also be read out by the core via data register LP\_TX.

If the Transmitter is disabled while performing writes to the transmit FIFO, a FIFO full condition will be signalled after just 2 writes.

Shadow registers have been provided for the transmit buffer registers. Using these shadow registers, both stages of the 2-deep FIFO may be read without updating the status registers. The LP\_TXIN\_SHDW register corresponds to the input stage of the FIFO and the LP\_TXOUT\_SHDW register corresponds to the output stage of the FIFO.



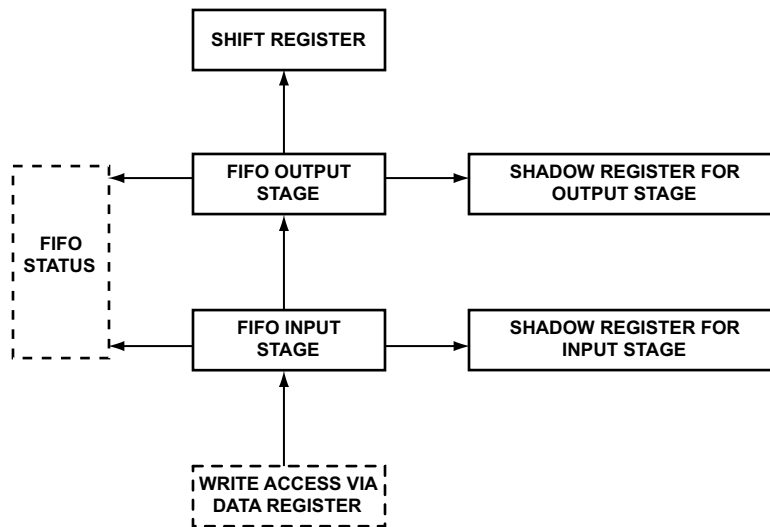


**Figure 28-7:** Transmit FIFO path

When a link port is configured as receiver, data is transferred to the core or DMA from the full 4-deep receive FIFO. An internal packing register performs the packing of data to 32 bits. Four reads can occur from the receive buffer by the core or DMA before it signals an empty condition. Status of the 4-deep read buffer FIFO is reflected by the `FFST` bits in the `LPSTAT` register. The core can access this FIFO through the `LP_TX` register.

**NOTE:** When receive FIFO overflows with Overflow bit in Status Register set, any further data from the transmitter shall be lost and only the data retained in the receive FIFO can be retrieved further.

The `LACK` output signal is driven low by the receiver once the first byte of data for the last but one empty slot (in the 4-deep FIFO) is received. This is to prevent data loss due to the transmitter starting transmission of the next word before the `LACK` signal reaches the transmitter (due to the larger delay in synchronization). This guarantees that even after allowing for the extra synchronization cycle in the transmitter and receiver, there is no overflow in the receive FIFO. *LACK Generation Based on Receive FIFO Status* (link port receive and transmit buffers) below shows how FIFO slots influence the acknowledge signal generation. The greyed section indicates the received data and the white section indicates the empty locations.

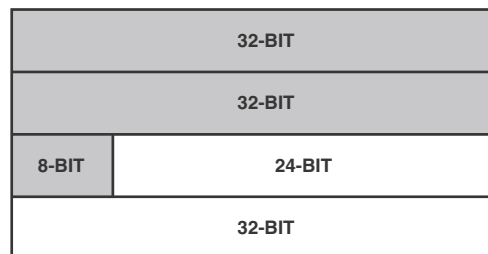


**Figure 28-8:** LACK Generation Based on Receive FIFO Status

**NOTE:** A 4-deep receive FIFO can be utilized only under a worst case scenario, and in all other cases, the FIFO must be thought of having only 3-deep stage.

Link port has memory mapped buffers for both receiver and transmit cases. Since these buffers are memory mapped, a JTAG based emulator (such as with ADI Cross Core Embedded Studio) may read the FIFO and thus can cause unexpected problems in data transfer. This can only happen during an emulation event (typically hitting a breakpoint or single-stepping), when emulator issues core reads via JTAG. CCES can block reading these FIFO buffers. For more information, please consult the CCES Help.

## Handshake for LP enable process

In a link port based system, the transmitter and the receiver may be enabled at different times. External pull downs should be used for LCLK and LACK.

If the receiver is enabled before the transmitter, LCLK (of the transmitter) is held low by the external pull down and the receiver is held off. Receiver can wait for a rising edge on the LCLK line to assert its Receive Service Request interrupt. This rising edge occurs only when transmitter starts driving the first data on to the bus, after it is enabled by the application.

If the transmitter is enabled before the receiver, LACK (of the receiver) is held low by the external pull down and transmission is held off, as shown in the below figure. Transmitter can wait for a rising edge on the LACK line to assert its Transmit Service Request interrupt. This rising edge is asserted as soon as receiver is enabled with LACK driven high subsequently by hardware.

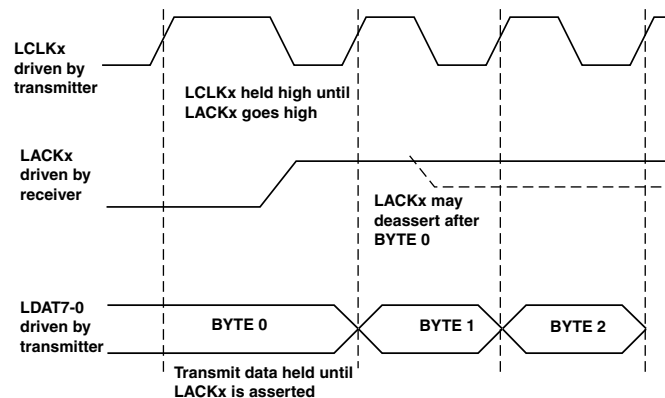


**Figure 28-9:** Enabling Transmitter before Receiver

**NOTE:** Service Request Interrupts / Status are asserted only when the link port (receiver or transmitter) is disabled.

## Clocking

Link port clock (LCLK) is derived out of the internal system clock (SCLK0). The link port clock to system clock ratio can be programmed using the DIV field of link port divider register (LPDIV). This programming is applicable only for the transmitter. The receiver can operate at any asynchronous frequency up to the maximum frequency, independent of the ratio programmed. The relationship between link port clock frequency, SCLK frequency, and the FSDIV value is expressed by the formula shown below.

If  $f_{LCLK}$  – Link clock frequency,  $f_{LCLK-MAX}$  – Link clock maximum frequency, and  $f_{SCLK}$  – System clock frequency, then:

- $f_{LCLK} = f_{SCLK} < \text{or} = f_{LCLK-MAX}$  if  $DIV = 0$
- $f_{LCLK} = f_{SCLK} / (2 * DIV)$  if  $DIV > 0$

While programming the  $LPDIV$  register to select the clock ratio, make sure that the LCLK frequency does not exceed the maximum frequency supported for the part (see data sheet). For example, if the SCLK frequency is 125 MHz and the data sheet limit for LCLK operation is 83 MHz,  $DIV$  should be greater than or equal to 1, so that LCLK frequency is less than or equal to 83 MHz.

## Multi-Processor Connectivity

The ADSP-BF60x has four link ports and each can operate independently, allowing glueless connection with up to four external processors. All link ports have dedicated DMA channels, allowing independent data transfers. The following figures are some examples of different bus connection topology that can be used in multi-processor system design. The inter-connection methods are not limited to these examples.

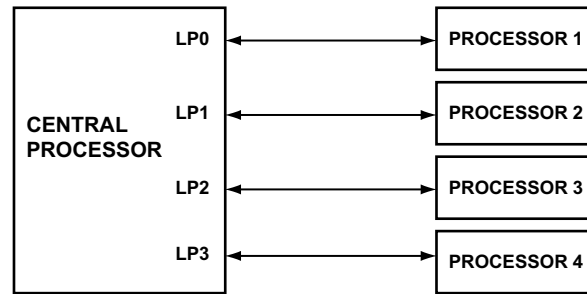


**Figure 28-10:** Central Processor Based Model

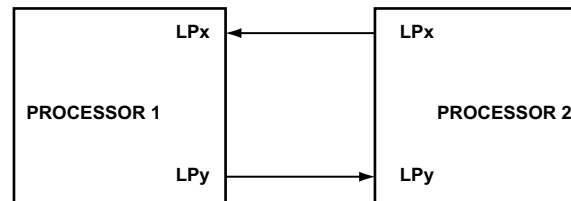


**Figure 28-11:** Link Port Full-duplex transfer Model

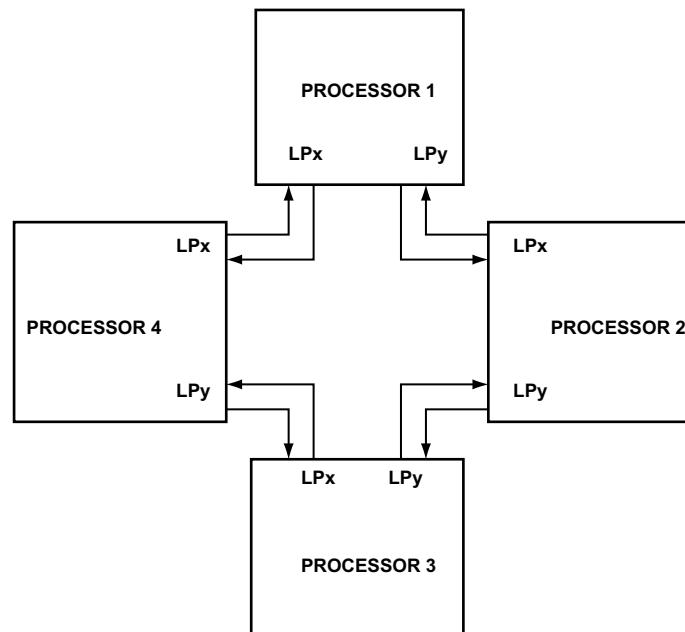


**Figure 28-12:** Link Port Ring Model

The link port protocol doesn't include built in support for multiple masters, although there may be situations where multiple devices try to become the bus master at the same time. Multi-master conflicts may be resolved using token passing. In token passing, the token is a software flag that passes between processors. At reset, the token is set to reside in the link port of one device, making it the master and the transmitter. When a receiver (slave) wants to become the master, it may assert its LACK line to get the master's attention. The master knows, through the software protocol, whether it is supposed to respond with actual data or whether it is being asked for the token. If the master wishes to give up the token, it may send back a user-defined token release word and thereafter clear its token flag. Simultaneously, the slave will set its token and can thereafter transmit. The token release word can be any user-defined value. Since both the transmitter and receiver are expecting a code word, this does not need to be exclusive of normal data transmission. If the master wishes to give up the token, it may send back a user-defined token release word and thereafter clear its token flag. Simultaneously, the slave examines the data sent back and if it is the token release word, the slave will set its token, and can thereafter transmit.

Link port includes handshake for informing enable instance to the other end of transfer (transmit or receive). However, it does not support handshakes to inform disable instance, while a chunk of data is being transferred. Application has to assume about the disabled state of the other end, and then take necessary actions. Similarly, there can be a situation in multi-processing environment where receiver did not read its full FIFO for a long time (owing to internal bus arbitrations etc). In such a case, transmitter may need some software / timer peripheral based time out to inform application that LACK is low for a long time.

## LP Operating Modes

The link port does not have particular modes of operation, as the peripheral is based on a simple protocol. The data transfer modes, Core based and DMA based are explained here.

## LP Data Transfer Modes

This section describes link port DMA and core data transfers.

### DMA Data Transfers

Dedicated DMA channels are available for each link port. DMA related activity is explained below.

1. **Data Receive:** Once the DMA channel and Link port module are configured and enabled, the external device begins writing data to the FIFO through the link port data pins. The FIFO detects this and in turn sends a DMA request. After the request is granted, the DMA transfer progresses until the FIFO is empty.
2. **Data Transmit:** Once the DMA channel and link port module are configured and enabled, setting the `LPEN` bit automatically asserts a DMA request when the TX FIFO is empty. After the request is granted, DMA transfer is performed filling the FIFO. The external device begins reading data from the FIFO through the link port data pins. The FIFO detects that there is room in the buffer and asserts another DMA request, continuing the process.

### Core Data Transfers

If DMA is disabled for a link port buffer, then the internal FIFO buffers may be written or read by the processor core as a memory-mapped register through the MMR access bus. In order to avoid FIFO overflow or underflow, the core should access the FIFO registers in one of the two following ways.

1. Access link port registers using an interrupt service routine (ISR) mapped to the link port data request interrupt. The link port interrupt request remains high only if the FIFO is accessible (if the FIFO is not full in transmit mode and not empty in receive mode).
2. Poll the FIFO status bits of `LPSTAT` register. Write to the Transmit FIFO if not full or read from the Receive FIFO if not empty.

## LP Event Control

This section describes how interrupts and status signals are used with the link ports.

## LP Interrupt Signals

Each link port has two dedicated interrupt lines registered with the System Event Controller: one Data Request interrupt and one Status interrupt. Data request interrupt is asserted with respect to FIFO conditions for data transfer and Status interrupt is asserted when Service Request status or Overflow status is set. Each of these interrupts are explained below.

- Data Request Interrupt.

This is asserted if FIFO is not full in transmission mode and FIFO is not empty in reception mode. This serves as a core triggered interrupt in non-DMA mode and as the DMA interrupt request in DMA mode. Generation of this interrupt is tied to the FFST (link port buffer status bit) in the LPSTAT register.

- Link Port Transmit Service Request Interrupt (LTRQ)

Link port service request interrupts allow a disabled link port to generate an interrupt when an external access is attempted. When a link port is configured as transmitter, Transmit Service Request Interrupt is enabled by setting the LTRQ\_MSK bit in the LPCTL register. When this bit is set, an external receiver can indicate to the disabled transmitter that it needs to receive data through the connected link port. The receiver does so by driving a high level on the LACK line. When the LACK of the disabled transmitter link port is detected high, a LTRQ interrupt is generated, and the transmitter can enable itself for data transfer with the receiver. Note that a pull down on LACK is required for proper function of this feature.

- Link Port Receive Service Request Interrupt (LRRQ)

When a link port is configured as receiver, Receive Service Request Interrupt is enabled by setting the LRRQ\_MSK bit in the LPCTL register. When set, an external transmitter can indicate to the disabled receiver that it needs to receive data through the connected link port. The transmitter does so by driving the first data out. When the LCLK of the disabled receiver link port is detected high, a LRRQ interrupt is generated, and the receiver can further enable itself for data transfer with the transmitter. Note that a pull down on the LCLK signal is required for proper function of this feature.

- Link Port Receive Overflow Interrupt (LPOVF)

This interrupt is generated when the receiver FIFO overflows and is enabled by setting the LPOVF\_MSK bit in the register. This may happen if the transmitter continues to transmit data even though the receiver has de asserted LACK thereby causing the receive FIFO to overflow.

## Enabling Link Port interrupts

A data request interrupt is fed to the System Event Controller directly and can be controlled separately from the application.

Service interrupts and the Overflow interrupt can be masked by setting the corresponding mask bits in LPCTL register, as these are OR'ed and fed to the SIC as a single LPSTAT interrupt. These interrupts are latched and stored in the associated bits of LPSTAT register. If an LPSTAT interrupt occurs, in the ISR, programs should read the LPSTAT register bits to determine the type of interrupt. Note that these bits are write-one-to-clear (W1C); writing one to the bit resets the bit and disables the corresponding interrupt.

## LP Status and Error Signals

This section explains the various status signals in the LPSTAT register.

- Transfer Status signals

Link port bus status (LPBS): This bit gives the status of bus busy/idle condition, when the link port is configured as transmitter. The LPBS is kept high if data is being driven by the link port in to the link port pins. Programs may poll this bit after polling the FFST bit to safely disable the link port.

Link buffer status (FFST): This field directly indicates the status of the FIFO (including empty / full conditions) during data transfer. Software can poll this field in link port status register before writing in to FIFO (in case of transmission) or reading from FIFO (in case of reception). The FFST bit is automatically cleared when the link port is disabled.

- Transfer Request Status signals

Link port receive request status (LRRQ) / link port transmit request status (LTRQ): These bits indicate that an external receiver wants to receive data (in case the link port is a disabled transmitter) or an external transmitter wants to send data (in case the link port is a disabled receiver). Software can poll this bit to enable the transmitter or receiver accordingly.

- Error Status signals

Link buffer error status (LERR): In receive mode 32-bit data is received in four chunks of 8-bit data. This is then packed to a single 32-bit data before loading the FIFO. The LERR bit is high during this packing process and goes low after packing.

Link port overflow status (LPOVF): This bit is set when the receive FIFO overflows. This may occur if the transmitter continues to transmit data even though the receiver has de asserted LACK thereby causing the receiver FIFO to overflow.

## LP Programming Model

The following sections provide information on configuring the operating mode and enabling the link ports.

- [\*Setting Up LP DMA Transmit Operation\*](#)
- [\*Setting Up LP DMA Receive Operation\*](#)
- [\*Setting Up LP Core Transmit Operation\*](#)
- [\*Setting Up LP Core Receive Operation\*](#)

## Setting Up LP DMA Transmit Operation

This section describes the steps for configuring the link ports in DMA transmit mode.

1. Enable the link port pins in the GPIO Port Mux using the appropriate Port FER and Port MUX registers.
2. Install interrupt handlers for DMA and for transfer status (Service Request Interrupt).
3. Configure the link port to transmit by setting the LP\_CTL bit and enable the transmit request interrupt mask by setting the TRQMSK bit.
4. Program the link port clock divider by writing a value to the LP\_DIV register.
5. If using DMA Stop mode/Auto Buffer mode, program the DMA registers.

*ADDITIONAL INFORMATION:* An example configuration is: Start Address, XCount, Xmodify and Config (Stop/Auto, Psize=1, Msize=4, Interrupt generation and Memory Read.

6. If using DMA Stop mode/Auto Buffer mode, program DMA Registers.  
*ADDITIONAL INFORMATION:* An example configuration is: Start Address, XCount, Xmodify and Config (Stop/Auto, Psize=1, Msize=4, Interrupt generation and Memory Read.
7. Wait for the link port receiver (connected externally) to be enabled. The application can wait for the transmit service request interrupt to assert.
8. Clear the transmit service request interrupt status by writing 1 to the LP\_STAT.LTRQ bit.
9. Enable DMA by setting the EN bit in the DMA configuration register.
10. Enable the link port by setting the LP\_CTL.EN bit.
11. Wait for DMA to assert transfer completion interrupt.
12. Clear the DMA interrupt source by writing 1 to the IRQDONE bit of the DMA status register.

## Setting Up LP DMA Receive Operation

This section describes steps for using the link ports in receive mode of DMA

1. Enable link port pins in GPIO Port Mux using the appropriate Port FER and Port MUX registers.
2. Install interrupt handlers for DMA and for transfer status (Service Request Interrupt).
3. Configure the link port for reception (clear the LP\_CTL.TRAN bit) and enable the receive request interrupt mask by setting the LP\_CTL.RRQMSK bit.



4. If using DMA Stop mode/Auto Buffer mode, program the DMA registers.

*ADDITIONAL INFORMATION:* An example configuration is: Start Address, XCount, Xmodify and Config (Stop/Auto, Psize=1, Msize=4, interrupt generation and memory write).

5. If using DMA Array mode/List mode, create DMA configuration data structures filled with components.

*ADDITIONAL INFORMATION:* An example configuration is: Start Address, XCount, XModify, Config (Array/List, Psize=1, Msize=4, Interrupt generation, Memory Write and Fetch=4/5) and Next Desc pointer (if List mode). Further program DMA configuration register (Array/List, Psize=1, Msize=4, Memory Write and Fetch=4/5) and program Next Desc pointer (if list mode).

6. Wait for the link port transmitter (connected externally) to be enabled with subsequent data transmission. The application can wait for Receive Service Request Interrupt to assert.
7. Clear the receive service request interrupt status by writing 1 to the LP\_STAT.LRRQ bit.
8. Enable DMA by setting the EN bit in the DMA configuration register.
9. Enable the link port by setting the LP\_CTL.EN bit.
10. Wait for DMA to assert the transfer complete interrupt.
11. Clear the DMA interrupt source by writing 1 to the IRQDONE bit of the DMA status register.

## Setting Up LP Core Transmit Operation

This section describes steps for using the link ports in processor core based transmission.

1. Enable the link port pins in the GPIO Port Mux using the appropriate Port FER and Port MUX registers.
2. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source/ID as the DMA interrupt line in the SEC.
3. Configure the link port for transmission by setting the LP\_CTL.TRAN bit) and enable the transmit request interrupt mask by setting the LP\_CTL.TRQMSK bit).
4. Program the link port clock divider by writing a value in to the LP\_DIV register.
5. Wait for the link port receiver (connected externally) to be enabled. The application can wait for a transmit service request interrupt to assert.
6. Clear the transmit service request interrupt status by writing 1 to the LP\_STAT.LTRQ bit.
7. Enable the link port by setting the LP\_CTL.EN bit.
8. The interrupt is asserted whenever there is free space in the FIFO. The application can write to the LP\_TX register based on the FIFO conditions (half or empty) reflected in the LP\_CTL.FFST bit field.

## Setting Up LP Core Receive Operation

This section describes steps for using the link ports in processor core based reception.

1. Enable the link port pins in the GPIO Port Mux using the appropriate Port FER and Port MUX registers.
2. Install interrupt handlers for data transfer and for transfer status (service request interrupt). The interrupt handlers for data transfer are the same source/ID as the DMA interrupt line in the SEC).
3. Configure link port for reception (clear LP\_CTL.TRAN bit) and enable receive request interrupt mask (set LP\_CTL.RRQMSK bit).
4. Wait for the link port transmit (connected externally) to be enabled with subsequent transmission of data. Application can wait for Receive Service Request Interrupt to be asserted.
5. Clear the receive service request interrupt status by writing 1 to the LP\_STAT.LRRQ bit.
6. Enable the link port by setting the LP\_CTL.EN bit.
7. The interrupt is asserted whenever there is free space in the FIFO. The application can read from the LP\_RX register based on the FIFO conditions (1 or 2 or 3 data available) which is reflected in the LP\_CTL.FFST bit field.

## LP Register Descriptions

Link Port (LP) contains the following registers.

**Table 28-5:** LP Module Registers

| Name          | Description                   |
|---------------|-------------------------------|
| LP_CTL        | Control Register              |
| LP_STAT       | Status Register               |
| LP_DIV        | Clock Divider Value           |
| LP_TX         | Transmit Buffer               |
| LP_RX         | Receive Buffer                |
| LP_TXIN_SHDW  | Shadow Input Transmit Buffer  |
| LP_TXOUT_SHDW | Shadow Output Transmit Buffer |

## Control Register

The LP\_CTL register provides LP interrupt masking, selection of transfer direction, and link port enable.

### LP\_CTL: Control Register - R/W

Reset = 0x0000 0000

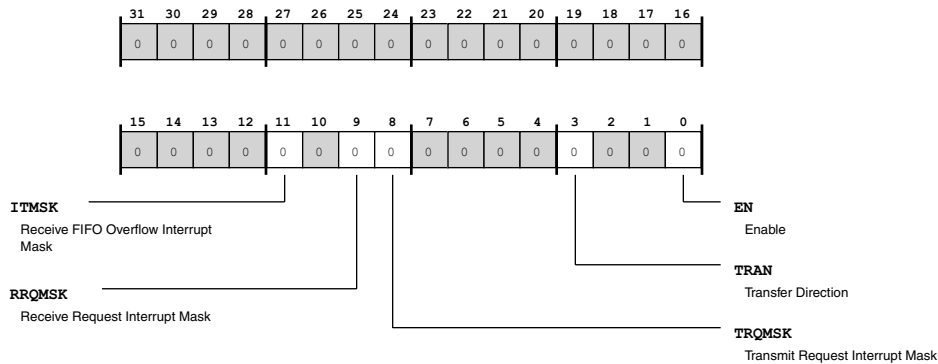


**Figure 28-13:** LP\_CTL Register Diagram

**Table 28-6:** LP\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                       |
|---------|----------|-------------------------------------------------------------------------------|
| 11      | ITMSK    | Receive FIFO Overflow Interrupt Mask.<br>Receive FIFO Overflow Interrupt Mask |
|         |          | 0 Mask<br>Disable Receive FIFO Overflow Interrupt                             |
|         |          | 1 Unmask<br>Enable Receive FIFO Overflow Interrupt                            |
| 9       | RRQMSK   | Receive Request Interrupt Mask.<br>Link Port Receive Request Mask             |
|         |          | 0 Mask<br>Disable Receive Request interrupt.                                  |
|         |          | 1 Unmask<br>Enable Receive Request interrupt.                                 |

**Table 28-6:** LP\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                          |                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|
| 8       | TRQMSK   | Transmit Request Interrupt Mask.<br>Link Port Transmit Request Mask                                                                                                                              |                                              |
|         |          | 0                                                                                                                                                                                                | Mask<br>Disable Transmit Request interrupt.  |
|         |          | 1                                                                                                                                                                                                | Unmask<br>Enable Transmit Request interrupt. |
| 3       | TRAN     | Transfer Direction.<br>The LP_CTL . TRAN bit selects the transfer direction as transmit (if set) or receive (if cleared) for link buffer.                                                        |                                              |
|         |          | 0                                                                                                                                                                                                | Receive<br>Direction transfer is receive     |
|         |          | 1                                                                                                                                                                                                | Transmit<br>Direction transfer is transmit   |
| 0       | EN       | Enable.<br>The LP_CTL . EN enables or disables the link buffer. When the processor disables the (LP_CTL . EN transitions from high to low), the processor clears the corresponding LP_STAT bits. |                                              |
|         |          | 0                                                                                                                                                                                                | Disable<br>Disable linkport                  |
|         |          | 1                                                                                                                                                                                                | Enable linkport<br>Enable linkport           |

## Status Register

The LP\_STAT register provides status information on link port interrupts, FIFO, buses, and receive/transmit requests.

**LP\_STAT: Status Register - R/W**

Reset = 0x0000 0000

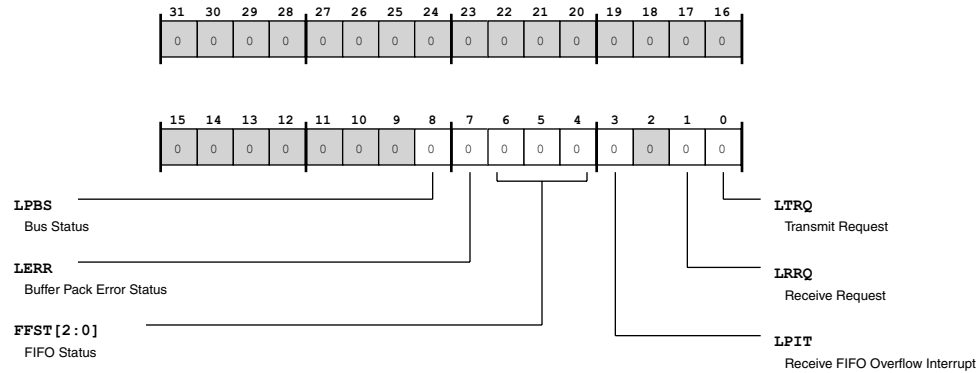


**Figure 28-14:** LP\_STAT Register Diagram

**Table 28-7:** LP\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | LPBS     | Bus Status.<br>The LP_STAT.LPBS bit indicates the LPDAT bus status.<br>LP_STAT.LPBS is kept high if data is being driven by the link port into the LP_DN pins.                                                                                                                                                                                                                                                                                                  |
|         |          | 0<br>Bus is Idle<br>Link Port Bus is idle                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1<br>Bus Busy<br>Link Port Bus is busy                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 7       | LERR     | Buffer Pack Error Status.<br>The LP_STAT.LERR bit indicates packing status.<br>In receive mode, 32-bit data is received in 4 blocks of 8-bit data. Then, the data is packed to get a single 32-bit data before loading the FIFO. The LP_STAT.LERR bit is high during this packing process and goes low after packing.<br>In transmit mode, 32-bit data in FIFO is unpacked to 4 blocks of 8-bit data before sending. The LP_STAT.LERR is high during unpacking. |
|         |          | 0<br>Packing Complete<br>Packing done                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 1<br>Packing Incomplete<br>Packing is in progress                                                                                                                                                                                                                                                                                                                                                                                                               |

**Table 28-7:** LP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                           |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6:4     | FFST     | FIFO Status.<br>The LP_STAT.FFST bits indicate the FIFO status. These bits are cleared when the LP is disabled.                                                                                                                   |
|         |          | 0 TX - Empty; RX -Empty<br>Link buffer (TX OR RX) empty                                                                                                                                                                           |
|         |          | 1 TX - reserved ; RX - One Word<br>RX has 1 word of data. TX reserved                                                                                                                                                             |
|         |          | 2 TX - reserved; RX - Two Word<br>RX has 2 word of data. TX reserved.                                                                                                                                                             |
|         |          | 3 TX - reserved; RX - Three Word<br>RX has 3 word of data. TX reserved.                                                                                                                                                           |
|         |          | 4 TX - One Word; RX - Four word<br>RX has 4 word of data. TX 1 word of data.                                                                                                                                                      |
|         |          | 5 TX - Reserved; RX - Reserved<br>RX reserved.. TX reserved.                                                                                                                                                                      |
|         |          | 6 TX - FIFO Full; RX - Reserved<br>RX reserved.. TX reserved.                                                                                                                                                                     |
|         |          | 7 TX - Reserved; RX - Reserved<br>RX reserved.. TX reserved.                                                                                                                                                                      |
| 3       | LPIT     | Receive FIFO Overflow Interrupt.<br>This interrupt is generated when the receiver FIFO overflows. This overflow may happen if the transmitter continues to transmit data even though the receiver has de-asserted the LP_ACK pin. |
| 1       | LRRQ     | Receive Request.<br>The LP generates this interrupt when the LP_CLK pin of a disabled link port (the receiver) is forced high by another link port (the transmitter).                                                             |
| 0       | LTRQ     | Transmit Request.<br>The LP generates this interrupt when the LP_ACK pin of a disabled link port (the transmitter) is forced high by another link port (the receiver).                                                            |

## Clock Divider Value

The LP\_DIV register selects the divisor for ratio between the internal LP clock (LCLK) and system clock (SCLK). This programming is applicable only for the transmitter. The receiver can operate at any asynchronous frequency up to the maximum frequency independent of the ratio programmed.

**LP\_DIV: Clock Divider Value - R/W**

Reset = 0x0000 0000

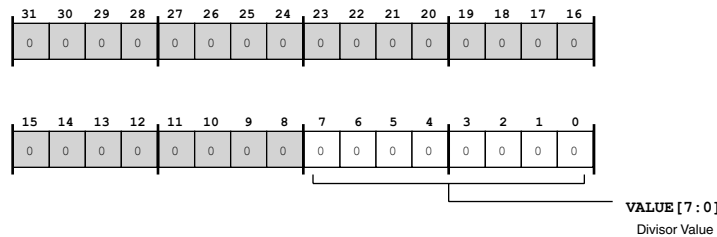


**Figure 28-15:** LP\_DIV Register Diagram

**Table 28-8:** LP\_DIV Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:0     | VALUE    | <p>Divisor Value.</p> <p>The LP_DIV.VALUE bits select the clock divider (relating the LP' internally generated clock (LCLK) to the system clock (SCLK). The LP_DIV.VALUE should be programmed prior to LP enable.</p> <p>For LP_DIV.VALUE = 0, LCLK = SCLK</p> <p>For LP_DIV.VALUE = xxxxxxxx, LCLK = SCLK / (2 x DIV)</p> |

## Transmit Buffer

The LP\_TX register buffers the transmit data flow through the LP. The transmit buffer is a two-location deep FIFO. In the transmit buffer, the input stage of the FIFO is used to accept core data or DMA data from internal memory, and the data is transferred to the ports from the output stage of the FIFO. The output stage performs the unpacking in the transmit buffer. The least significant byte is transmitted first. As each word is unpacked and transmitted, the next location in FIFO becomes available and a new DMA request is made if DMA is enabled. If the register becomes empty, the LP asserts the LP\_CLK signal. For more information on LP buffer features and operations, see the LP functional description.

**LP\_TX: Transmit Buffer - R/W**

Reset = 0x0000 0000

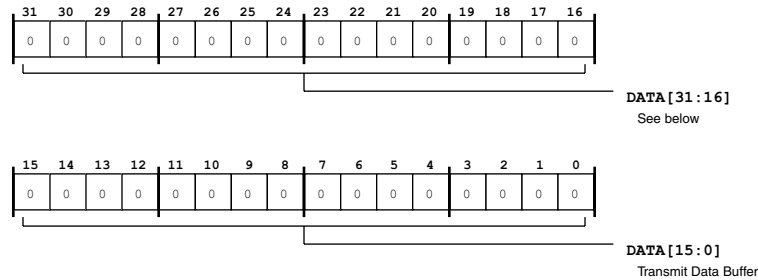


**Figure 28-16:** LP\_TX Register Diagram

**Table 28-9:** LP\_TX Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | DATA     | Transmit Data Buffer.   |

**Receive Buffer**

The LP\_RX register buffers the receive data flow through the LP. The receive buffer is a four-location deep FIFO. In the receive buffer, data is transferred to the core or DMA from the receive FIFO where an internal register does the packing. This packing register is not software accessible. For more information on LP buffer features and operations, see the LP functional description.

**LP\_RX: Receive Buffer - R/W**

Reset = 0x0000 0000

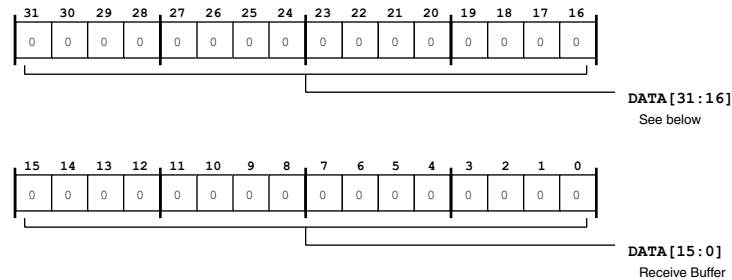


**Figure 28-17:** LP\_RX Register Diagram



**Table 28-10:** LP\_RX Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | DATA     | Receive Buffer.         |

## Shadow Input Transmit Buffer

The LP\_TXIN\_SHDW register contains the same data as the input stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP\_STAT register.

**LP\_TXIN\_SHDW: Shadow Input Transmit Buffer - R/NW**

Reset = 0x0000 0000

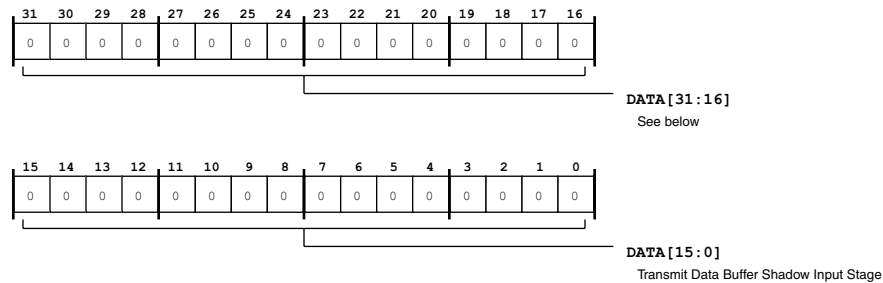


**Figure 28-18:** LP\_TXIN\_SHDW Register Diagram

**Table 28-11:** LP\_TXIN\_SHDW Register Fields

| Bit No. | Bit Name | Description/Enumeration                  |
|---------|----------|------------------------------------------|
| 31:0    | DATA     | Transmit Data Buffer Shadow Input Stage. |

## Shadow Output Transmit Buffer

The LP\_TXOUT\_SHDW register contains the same data as the output stage of the transmit buffer. Read of this shadow transmit buffer does not update the LP\_STAT register.

**LP\_TXOUT\_SHDW: Shadow Output Transmit Buffer - R/NW**

Reset = 0x0000 0000

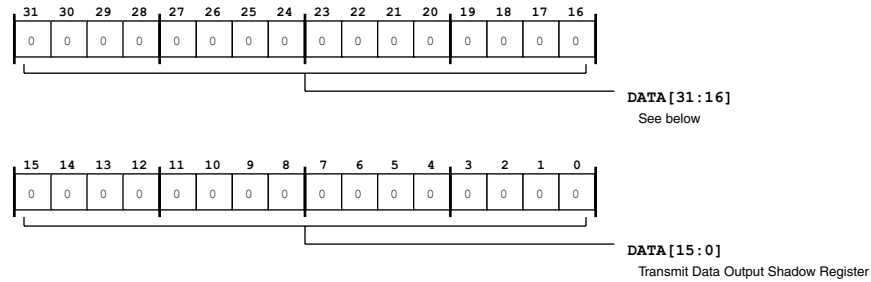


**Figure 28-19:** LP\_TXOUT\_SHDW Register Diagram

**Table 28-12:** LP\_TXOUT\_SHDW Register Fields

| Bit No. | Bit Name | Description/Enumeration               |
|---------|----------|---------------------------------------|
| 31:0    | DATA     | Transmit Data Output Shadow Register. |

## 29 Video Subsystem (VID)

The ADSP-BF60x features a sophisticated video subsystem (VID) consisting of the following functional blocks:

- Three Video Ports (PPI)
- The Pixel Compositor (PIXC)
- The Pipelined Vision Processor (PVP)
- The Video Interconnect
- 18 DMA Channels

The enhanced parallel peripheral interfaces (EPPIs) can operate in video input and video output modes, and also support several general-purpose modes of operation. The EPPIs are aware of video frame synchronization, blanking concepts and color formats. They can receive up to 16-bit video data directly from video sensors (cameras) and also directly control displays on the output.

The pixel compositor (PIXC) supports color space conversion and alpha blending for video overlays.

The pipelined vision processor (PVP) is a framework for various vision processing elements, which are targeting mainly edge and object detection strategies.

The video interconnect is a local, distributed bus system, which interconnects the EPPI ports, the PIXC and the PVP.

### VID Features

The following is a brief list of the features of the video subsystem.

- Bandwidth-saving pre-processing and post-processing on video inputs and outputs
- Video analytics and color-space conversion during video reception
- Alpha blending during video transmission
- Concurrent memory-to memory co-processing

### VID Functional Description

The following sections provide a functional description of the VID.

## VID Register List

The VID selects connections for the PVP and PPI modules, configuring connections to use these peripherals for video operations. Also, the VID selects broadcast or non-broadcast mode for handling the receive Y channel data. Note that these connections are static and have to be configured before enabling the PPI, PIXC, and PVP blocks. A set of registers govern VID operations. For more information on VID functionality, see the VID register descriptions.

**Table 29-1:** VID Module Registers

| Name     | Description                      |
|----------|----------------------------------|
| VID_CONN | Video Subsystem Connect Register |

## VID Block Diagram

The [VID Block Diagram](#) shows an overview of the video subsystem including the PIXC, PVP, and EPPI.

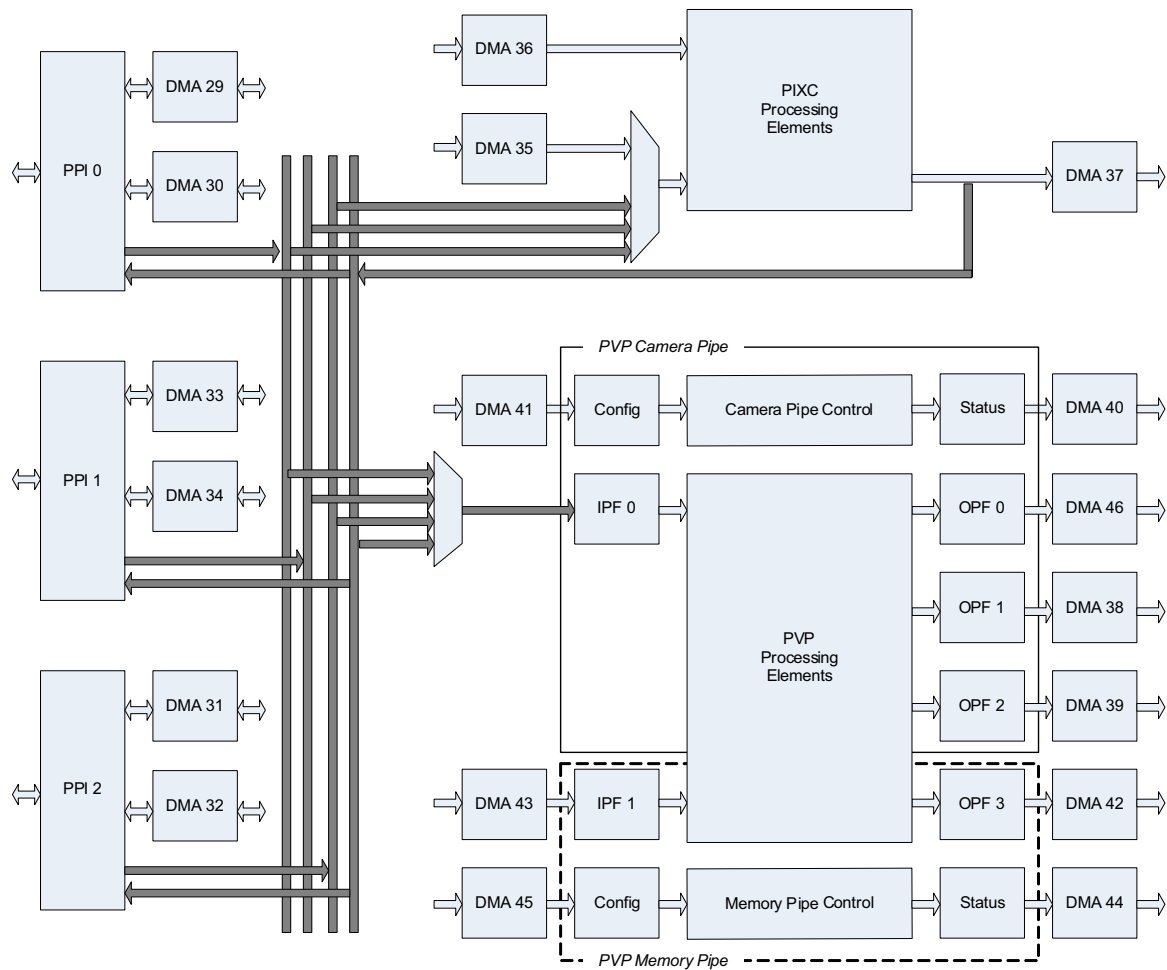


**Figure 29-1:** VID Block Diagram

## VID Architectural Concepts

Traditionally, the three EPPIs operate in DMA mode and as such mainly transport data between external world and on-chip L1 and/or L2 SRAM or external L3 memories. Each EPPI interface contains two DMA channels. The primary one is used for general data or alternatively only for the luminance component. The secondary can be optionally used for chrominance.

## PIXC and PVP Co-Processing

The PIXC and PVP processors can operate in coprocessor mode. In this mode, the two blocks receive input data from system memory via dedicated input DMA channels, process it, and then store the results back to system memory using dedicated output DMA channels. For this memory-to-memory operation, the PIXC features two input DMA channels, one for the background image and one for the overlay, and also an output DMA channel to store the results back to memory. Similarly, the PVP has a DMA channel pair of input and output for memory-to-memory operation. This concept is referred to as the camera pipe.

Before data is processed by the PVP processing elements, it can be preconditioned by the input formatter IPF1. Results can be post-conditioned by the output formatter OPF3.

The following figure illustrates the coprocessor architecture. First video data is received by one of the EPPI interfaces and stored into the system memory. Then, the PIXC and PVP processors have access to the data as the Blackfin cores do. All processing units may process the data either in parallel or serial fashion.

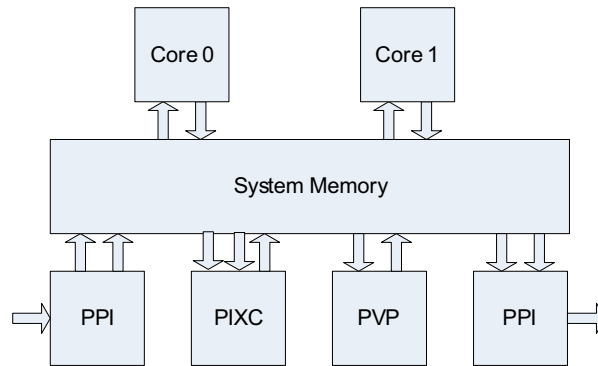


**Figure 29-2:** Coprocessor Topology

When visualizing a typical signal flow in coprocessor topology as in [Coprocesor Topology](#), a few characteristics of this topology become obvious. On the positive side, each block can be used multiple times (time domain multiplexing) for different jobs when processing one set of data. There is perfect flexibility in order of operations. However, fundamental rules of multi-processing apply—processing threads have to be synchronized when an operation is dependent on the results of operations performed by any other processing unit. Furthermore, each hand-over is performed through system memory. This requires a significant amount of memory as well as substantial memory bandwidth.

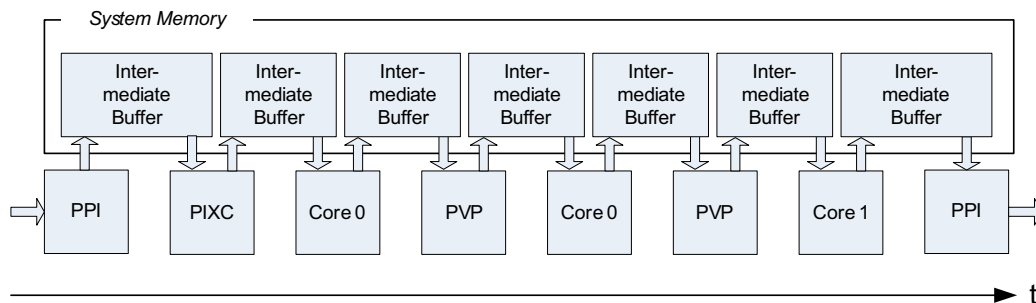


**Figure 29-3:** Coprocessor Signal Flow Example

## PIXC and PVP Preprocessing

The PIXC and PVP processors can also operate in preprocessing mode in order to conserve memory bandwidth and reduce power dissipation.

When in receive mode, each EPPI can perform data broadcasting on its main (luminance) channel. Received data is forwarded to the DMAs and the video interconnect simultaneously. The PIXC and the

PVP can receive video data from the video interconnect. This way, the PIXC and/or PVP can preprocess received video data before DMA stores it to memory.

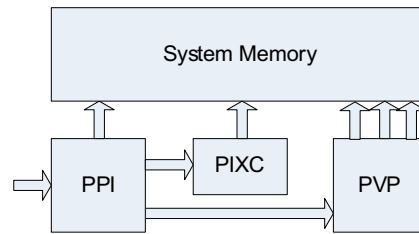


**Figure 29-4:** EPPI Data Broadcast to DMA, PIXC, and/or PVP

*EPPI Data Broadcast to DMA, PIXC, and/or PVP* illustrates that a receiving EPPI can broadcast data to up to three destinations: to the memory via DMA, to the PIXC and the PVP via video interconnect. When the PIXC receives data from any of the three EPPIs it operates in color space conversion mode and its overlay DMA input is not used.

Neither direct EPPI-to-EPPI operation nor EPPI-to-PIXC-to-EPPI operation is supported. However, the PIXC and PVP can operate in the same pipe as shown in the figure below.

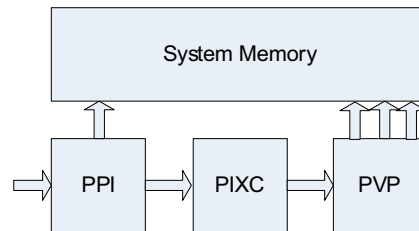


**Figure 29-5:** PPI, PIXC and PVP Chaining in Preprocessing Pipe

This mode is useful when the EPPI receives color data in RGB format. The PIXC can transform it to YCrCb format so that the PVP finally processes the Y in monochrome fashion. The PIXC does not support data broadcasting to both DMA and video interconnect. Either the PIXC output is written to memory or forwarded to the PVP.

When the PVP receives data from the EPPIs or the PIXC through the video interconnect, the data can be processed by up to three camera pipes. Data is preconditioned by the input formatter IPF0. Similarly, the result of each camera pipe can be post-conditioned by the respective output formatters OPF0, OPF1 and OPF2 and is then written to system memory via dedicated DMA channels.

Whenever the PIXC and PVP operate in a common pipeline, the data must be sourced by one of the EPPIs. Whenever a EPPI is used as the source for a PIXC and/or PVP pipe, the respective EPPI must operate in external clock mode.

## PIXC Post Processing

The pixel compositor can perform post-processing. In this mode, video frames residing in system memory can be transmitted via DMA, while the PIXC applies an overlay and can convert the color space before the resulting data is forwarded to any of the EPPIs via video interconnect.

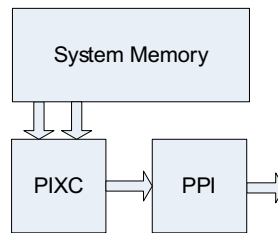


**Figure 29-6:** PIXC and PPI Chaining in Post-processing Pipe

When in post-processing mode, the PIXC must receive data from the DMAs, it cannot receive data from another EPPI through video interconnect.

## Concurrency of Pre-, Co- and Post-Processing

Pre-, co- and post-processing are not mutually exclusive concepts. In general a mix of all three topologies is often the best strategy for applications. The following figure shows an example where the PVP's camera pipes operate in preprocessing mode, its memory pipe operates in co-processing mode, and the finally the PIXC does post-processing.

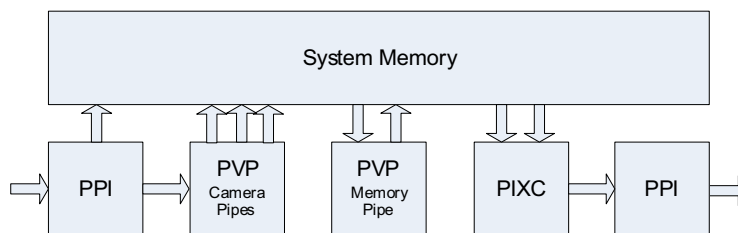


**Figure 29-7:** Concurrency of Pre-, Co- and Post-Processing

The pre- and post-processing elements typically operate in a static or pseudo static configuration. Certainly, the various coefficients, like filter settings and overlay regions may dynamically change from frame to frame. The basic data flow and operation mode, however, tends to be static over time or over at least a period of time. The co-processing elements are different in nature, they typically perform a job in any desired mode and are then reconfigured for a complete different job (time domain multiplexed). The PVP memory pipe has been designed to perform complete job list without support required from the Blackfin cores.

## VID Status and Error Signals

The video subsystem does not have dedicated status or error interrupts. However, the VID signals errors to the associated EPPI peripheral when stall or overflow conditions occur. While the PVP has been designed to never stall on the video interconnect, the PIXC may do so. This is reported by the `PxPERR` flag to the `PPIX_STAT` register. If the PIXC and the PVP are configured to receive data from the same EPPI input, the error flag only signals the stalling condition of the PIXC. The PVP continues to get new data.



## VID Programming Model

The video subsystem is typically configured in static or pseudo-static fashion. There is only one MMR, the `VID_CONN` register, which needs to be written before the EPPI, PIXC and PVP can be enabled. When programs must alter `VID_CONN` at run time, it is important that all impacted video blocks are disabled.

The `VID_CONN.PIXCOIN` bit field controls whether the PIXC receives input from DMA or from a designated EPPI port. Similarly, the `VID_CONN.PVPOIN` bit field determines the data source of the PVP. When altering the `VID_CONN.PIXCOIN` bit field, the PIXC and both the former and the newly selected EPPIs need to be disabled. However, the other PPI and the PVP may continue operation without being disturbed.

For EPPIs operating in transmit mode, the three `VID_CONN.PPIxTX` bit fields control whether the respective PPI module receives data from DMA channel or from the PIXC. Only out of the PPIs or the PVP or the DMA can receive data from PIXC at a time.

For EPPIs operating in receive mode, the three `VID_CONN.PPIxBCST` bits enable broadcasting of the main (luminance) input data. The bits only need to be set (=1) if received data is broadcast to both the video interconnect and DMA system. If not set (=0), the `VID_CONN.PIXCOIN` and `VID_CONN.PVPOIN` bits determine whether data goes to either the video interconnect or to the DMA system. Broadcasting only takes place if indeed at least one of `VID_CONN.PIXCOIN` or `VID_CONN.PVPOIN` bit fields instruct the modules to consume data from the respective EPPI.

Care must be taken to ensure that the configuration of the EPPIs, PIXC and PVP works with the VID settings in a meaningful way. Also, if significant mode switching takes place on one module, the other modules shall always be temporarily disabled.

## VID Performance

The maximum EPPI frequency is 83 MHz. Assuming each clock contains a valid pixel, the maximum pixel rate is 83 MPixels/s. The same limitation also applies to the PVP's memory pipe. Unless the base clock (SCLK0) is lower than 83 MHz, memory pipe throughput is  $SCLK0/2$ . The PIXC can operate at full SCLK0 speed in memory-to-memory mode. The bandwidth consumed by memory-to-memory operations of the PVP and the PIXC can be controlled via the bandwidth control functionality of the respective DMA channels. Similarly, the consumed bandwidth can be monitored by the bandwidth monitor functionality of the DMA channels.

## VID Register Descriptions

Video Subsystem Registers (VID) contains the following registers.

Table 29-2: VID Module Registers

| Name     | Description                      |
|----------|----------------------------------|
| VID_CONN | Video Subsystem Connect Register |

Video Subsystem Connect Register

The VID\_CONN selects connections for the PVP and PPI modules, configuring connections to use these peripherals for video operations. Also, the VID\_CONN selects broadcast or non-broadcast mode for handling the receive Y channel data. Note that these connections are static and have to be configured before enabling the PPI, PIXC, and PVP blocks.

VID\_CONN: Video Subsystem Connect Register - R/W

Reset = 0x0000 0000

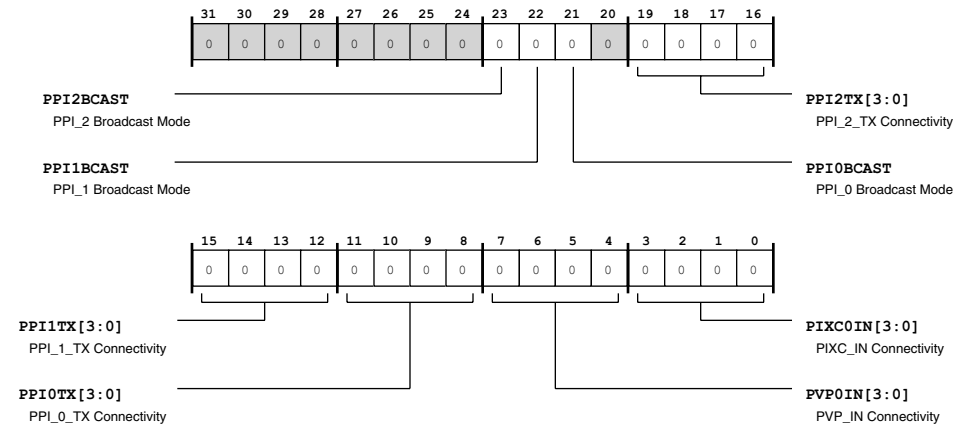


Figure 29-8: VID\_CONN Register Diagram

Table 29-3: VID\_CONN Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                               |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23      | PPI2BCAST | PPI_2 Broadcast Mode.<br>The VID_CONN.PPI2BCAST selects broadcast or non-broadcast mode for receive Y channel data. For more information, see the VID_CONN.PPI0BCAST bit description. |
|         |           | 0 Non-broadcast mode                                                                                                                                                                  |
|         |           | 1 Broadcast mode                                                                                                                                                                      |

**Table 29-3:** VID\_CONN Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 22      | PPI1BCAST | PPI_1 Broadcast Mode.<br>The VID_CONN.PPI1BCAST selects broadcast or non-broadcast mode for receive Y channel data. For more information, see the VID_CONN.PPIOBCAST bit description.                                                                                                                                                                                                                                                                                                                                                 |
|         |           | 0 Non-broadcast mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |           | 1 Broadcast mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 21      | PPI0BCAST | PPI_0 Broadcast Mode.<br>The VID_CONN.PPIOBCAST selects broadcast or non-broadcast mode for receive Y channel data. This bit is valid only when PPI_0 is in receive mode, and either VID_CONN.PIXCOIN = 0x1 or VID_CONN.PVPOIN = 0x1. In broadcast mode, the receive Y channel data is send on both the PxP interface and the PPI_0 DMA channel 0. In non-broadcast mode, the receive Y channel data is sent to either the PxP interface or the PPI_0 DMA channel based on the VID_CONN.PIXCOIN and VID_CONN.PVPOIN bits programming. |
|         |           | 0 Non-broadcast mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |           | 1 Broadcast mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 19:16   | PPI2TX    | PPI_2_TX Connectivity.<br>The VID_CONN.PPI2TX selects connection options for PPI_2 module output. These bits are ignored when PPI_2 is not in transmit mode.                                                                                                                                                                                                                                                                                                                                                                          |
|         |           | 0 PPI_2_TX to PPI_DMA<br>PPI_2_TX is connected to the PPI DMA engine (PPI_0 DMA Channel 0)                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|         |           | 1 PPI_2_TX to PIXC<br>PPI_2_TX is connected to PIXC output via PxP                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Table 29-3:** VID\_CONN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                      |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:12   | PPI1TX   | PPI_1_TX Connectivity.<br>The VID_CONN.PPI1TX selects connection options for PPI_1 module output. These bits are ignored when PPI_1 is not in transmit mode. |
|         |          | 0<br>PPI_1_TX to PPI_DMA<br>PPI_1_TX is connected to the PPI DMA engine (PPI_1 DMA Channel 0)                                                                |
|         |          | 1<br>PPI_1_TX to PIXC<br>PPI_1_TX is connected to PIXC output via PxP                                                                                        |
| 11:8    | PPI0TX   | PPI_0_TX Connectivity.<br>The VID_CONN.PPI0TX selects connection options for PPI_0 module output. These bits are ignored when PPI_0 is not in transmit mode. |
|         |          | 0<br>PPI_0_TX to PPI_DMA<br>PPI_0_TX is connected to the PPI DMA engine (PPI_0 DMA Channel 0)                                                                |
|         |          | 1<br>PPI_0_TX to PIXC<br>PPI_0_TX is connected to PIXC output via PxP                                                                                        |

**Table 29-3:** VID\_CONN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------|
| 7:4     | PVP0IN   | PVP_IN Connectivity.<br>The VID_CONN.PVP0IN selects connection options for PVP module input.                     |
|         |          | 0<br>PVP_IN to PVP_DMA<br>PVP data input is connected to the PVP DMA engine                                      |
|         |          | 1<br>PVP_IN to PPI_0_RX<br>PVP input is connected to PPI_0_RX via PxP (valid only when PPI_0 is in receive mode) |
|         |          | 2<br>PVP_IN to PPI_1_RX<br>PVP input is connected to PPI_1_RX via PxP (valid only when PPI_1 is in receive mode) |
|         |          | 3<br>PVP_IN to PPI_2_RX<br>PVP input is connected to PPI_2_RX via PxP (valid only when PPI_2 is in receive mode) |
|         |          | 4<br>PVP_IN to PIXC_OUT<br>PVP input is connected to PIXC output via PxP                                         |

**Table 29-3:** VID\_CONN Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                  |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------|
| 3:0     | PIXC0IN  | PIXC_IN Connectivity.<br>The VID_CONN.PIXC0IN selects connection options for PIXC module input.                          |
|         |          | 0<br>PIXC_IN to PIXC_DMA<br>PIXC image input is connected to the PIXC DMA IFIFO Channel (PIXC DMA channel 0)             |
|         |          | 1<br>PIXC_IN to PPI_0_RX<br>PIXC image input is connected to PPI_0_RX via PxP (valid only when PPI_0 is in receive mode) |
|         |          | 2<br>PIXC_IN to PPI_1_RX<br>PIXC image input is connected to PPI_1_RX via PxP (valid only when PPI_1 is in receive mode) |
|         |          | 3<br>PIXC_IN to PPI_2_RX<br>PIXC image input is connected to PPI_2_RX via PxP (valid only when PPI_2 is in receive mode) |

## 30 Pipelined Vision Processor (PVP)

The Pipelined Vision Processor (PVP) provides a set of 12 high-performance signal processing blocks that can be flexibly combined to form streaming data processing pipes. These blocks are optimized for tasks typical of video and image processing, analytics (e.g. advanced driver assistance systems), robotics, and 2-dimensional vector applications. The PVP works in conjunction with the processor core(s). It is optimized for convolution and wavelet-based object detection, classification, tracking, and verification algorithms. The PVP bundles a set of processing blocks required for high-speed 2-dimensional digital signal processing.

Most blocks are optimized for 2-dimensional video analytics operations and can be re-purposed for general-purpose operations. Other blocks have general-purpose functionality and might be used in a variety of applications outside of video processing.

### PVP Features

The PVP contains a number of highly configurable blocks that provide a broad set of pixel processing features:

- Variety of 1<sup>st</sup>-derivative and 2<sup>nd</sup>-derivative edge detection and classification methods
- Quad 5x5 convolution kernels with 8300 MMACS total
- Integral of pixels, variances, magnitudes and gradients
- Thresholds and histograms
- General-purpose 32-bit divider, multiplier, adder, and accumulator
- Up to 1280 x 960 pixel progressive video
- Up to 83M Pixels/second on each of up to four pipes
- Bandwidth-saving pre-processor architecture with data broadcasting to three “camera pipes”
- Memory-to-memory co-processing with autonomous sequencing of job lists
- Concurrent pre- and co-processing autonomous update of coefficients at frame boundaries
- Autonomous update of coefficients at frame boundaries

## PVP Functional Description

The Pipelined Vision Processor consists of 12 signal processing blocks, two input formatters and four output formatters. In combination and for simplicity, these elements are referred to as processing blocks.

The following sections describe all of the blocks that make up the PVP module. It provides details on their operation and interaction with each other and the processor core.

## PVP Register List

The pipelined vision processor PVP implements the signal and image processing algorithms that are required for pre-processing of video frames in advanced driver assistance systems (ADAS) applications. A number of processing blocks contribute to PVP operations. These include:

- Convolution/Down-scaling engine (CNV)
- Pixel Magnitude and Angle computation unit (PMA)
- Threshold, Histogram and Compression engine (THC)
- Arithmetic Computation Unit (ACU)
- Pixel Edge Classifier (PEC)
- Integral Image computation (IIM)
- Up-Down Scaler (UDS)

In addition to these processing blocks, the PVP also has a number of blocks that permit interfacing to other modules ( , ) and provide control for PVP operations:

- Input data formatter ( IPF )
- Output data formatter (OPF)
- Image pipe controller (IPC)

A set of registers govern PVP operations. For more information on PVP functionality, see the PVP register descriptions.

**Table 30-1:** PVP Module Registers

| Name      | Description      |
|-----------|------------------|
| PVP_REVID | Revision ID      |
| PVP_CTL   | Control          |
| PVP_IMSKn | Interrupt Mask n |



**Table 30-1:** PVP Module Registers (Continued)

| Name               | Description                        |
|--------------------|------------------------------------|
| PVP_STAT           | Status                             |
| PVP_ILAT           | Interrupt Latch Status n           |
| PVP_IREQn          | Interrupt Request n                |
| PVP_OPFn_CFG       | OPFn (Camera Pipe) Configuration   |
| PVP_OPFn_CTL       | OPFn (Camera Pipe) Control         |
| PVP_OPF3_CFG       | OPF3 (Memory Pipe) Configuration   |
| PVP_OPF3_CTL       | OPF3 (Memory Pipe) Control         |
| PVP_PEC_CFG        | PEC Configuration                  |
| PVP_PEC_CTL        | PEC Control                        |
| PVP_PEC_D1TH0      | PEC Lower Hysteresis Threshold     |
| PVP_PEC_D1TH1      | PEC Upper Hysteresis Threshold     |
| PVP_PEC_D2TH0      | PEC Weak Zero Crossing Threshold   |
| PVP_PEC_D2TH1      | PEC Strong Zero Crossing Threshold |
| PVP_IIMn_CFG       | IIMn Configuration                 |
| PVP_IIMn_CTL       | IIMn Control                       |
| PVP_IIMn_SCALE     | IIMn Scaling Values                |
| PVP_IIMn_SOVF_STAT | IIMn Signed Overflow Status        |
| PVP_IIMn_UOVF_STAT | IIMn Unsigned Overflow Status      |
| PVP_ACU_CFG        | ACU Configuration                  |
| PVP_ACU_CTL        | ACU Control                        |
| PVP_ACU_OFFSET     | ACU SUM Constant                   |
| PVP_ACU_FACTOR     | ACU PROD Constant                  |

**Table 30-1:** PVP Module Registers (Continued)

| Name              | Description                                |
|-------------------|--------------------------------------------|
| PVP_ACU_SHIFT     | ACU Shift Constant                         |
| PVP_ACU_MIN       | ACU Lower Sat Threshold Min                |
| PVP_ACU_MAX       | ACU Upper Sat Threshold Max                |
| PVP_UDS_CFG       | UDS Configuration                          |
| PVP_UDS_CTL       | UDS Control                                |
| PVP_UDS_OHCNT     | UDS Output HCNT                            |
| PVP_UDS_OVCNT     | UDS Output VCNT                            |
| PVP_UDS_HAVG      | UDS HAVG                                   |
| PVP_UDS_VAVG      | UDS VAVG                                   |
| PVP_IPF0_CFG      | IPF0 (Camera Pipe) Configuration           |
| PVP_IPFn_PIPECTL  | IPFn (Camera/Memory Pipe) Pipe Control     |
| PVP_IPFn_CTL      | IPFn (Camera/Memory Pipe) Control          |
| PVP_IPFn_TAG      | IPFn (Camera/Memory Pipe) TAG Value        |
| PVP_IPFn_FCNT     | IPFn (Camera/Memory Pipe) Frame Count      |
| PVP_IPFn_HCNT     | IPFn (Camera/Memory Pipe) Horizontal Count |
| PVP_IPFn_VCNT     | IPFn (Camera/Memory Pipe) Vertical Count   |
| PVP_IPF0_HPOS     | IPF0 (Camera Pipe) Horizontal Position     |
| PVP_IPF0_VPOS     | IPF0 (Camera Pipe) Vertical Position       |
| PVP_IPFn_TAG_STAT | IPFn (Camera/Memory Pipe) TAG Status       |
| PVP_IPF1_CFG      | IPF1 (Memory Pipe) Configuration           |
| PVP_CNVn_CFG      | CNVn Configuration                         |
| PVP_CNVn_CTL      | CNVn Control                               |

**Table 30-1:** PVP Module Registers (Continued)

| Name             | Description                   |
|------------------|-------------------------------|
| PVP_CNVn_C00C01  | CNVn Coefficients 0,0 and 0,1 |
| PVP_CNVn_C02C03  | CNVn Coefficients 0,2 and 0,3 |
| PVP_CNVn_C04     | CNVn Coefficient 0,4          |
| PVP_CNVn_C10C11  | CNVn Coefficients 1,0 and 1,1 |
| PVP_CNVn_C12C13  | CNVn Coefficients 1,2 and 1,3 |
| PVP_CNVn_C14     | CNVn Coefficient 1,4          |
| PVP_CNVn_C20C21  | CNVn Coefficients 2,0 and 2,1 |
| PVP_CNVn_C22C23  | CNVn Coefficients 2,2 and 2,3 |
| PVP_CNVn_C24     | CNVn Coefficient 2,4          |
| PVP_CNVn_C30C31  | CNVn Coefficients 3,0 and 3,1 |
| PVP_CNVn_C32C33  | CNVn Coefficients 3,2 and 3,3 |
| PVP_CNVn_C34     | CNVn Coefficient 3,4          |
| PVP_CNVn_C40C41  | CNVn Coefficients 4,0 and 4,1 |
| PVP_CNVn_C42C43  | CNVn Coefficients 4,2 and 4,3 |
| PVP_CNVn_C44     | CNVn Coefficient 4,4          |
| PVP_CNVn_SCALE   | CNVn Scaling Factor           |
| PVP_THCn_CFG     | THCn Configuration            |
| PVP_THCn_CTL     | THCn Control                  |
| PVP_THCn_HFCNT   | THCn Histogram Frame Count    |
| PVP_THCn_RMAXREP | THCn Max RLE Reports          |
| PVP_THCn_CMINVAL | THCn Min Clip Value           |
| PVP_THCn_CMINTH  | THCn Clip Min Threshold       |

**Table 30-1:** PVP Module Registers (Continued)

| Name             | Description                        |
|------------------|------------------------------------|
| PVP_THCn_CMAXTH  | THCn Clip Max Threshold            |
| PVP_THCn_CMAXVAL | THCn Max Clip Value                |
| PVP_THCn_TH0     | THCn Threshold Value 0             |
| PVP_THCn_TH1     | THCn Threshold Value 1             |
| PVP_THCn_TH2     | THCn Threshold Value 2             |
| PVP_THCn_TH3     | THCn Threshold Value 3             |
| PVP_THCn_TH4     | THCn Threshold Value 4             |
| PVP_THCn_TH5     | THCn Threshold Value 5             |
| PVP_THCn_TH6     | THCn Threshold Value 6             |
| PVP_THCn_TH7     | THCn Threshold Value 7             |
| PVP_THCn_TH8     | THCn Threshold Value 8             |
| PVP_THCn_TH9     | THCn Threshold Value 9             |
| PVP_THCn_TH10    | THCn Threshold Value 10            |
| PVP_THCn_TH11    | THCn Threshold Value 11            |
| PVP_THCn_TH12    | THCn Threshold Value 12            |
| PVP_THCn_TH13    | THCn Threshold Value 13            |
| PVP_THCn_TH14    | THCn Threshold Value 14            |
| PVP_THCn_TH15    | THCn Threshold Value 15            |
| PVP_THCn_HHPOS   | THCn Histogram Horizontal Position |
| PVP_THCn_HVPOS   | THCn Histogram Vertical Position   |
| PVP_THCn_HHCNT   | THCn Histogram Horizontal Count    |
| PVP_THCn_HVCNT   | THCn Histogram Vertical Count      |

**Table 30-1:** PVP Module Registers (Continued)

| Name                 | Description                       |
|----------------------|-----------------------------------|
| PVP_THCn_RHPOS       | THCn RLE Horizontal Position      |
| PVP_THCn_RVPOS       | THCn RLE Vertical Position        |
| PVP_THCn_RHCNT       | THCn RLE Horizontal Count         |
| PVP_THCn_RVCNT       | THCn RLE Vertical Count           |
| PVP_THCn_HFCNT_STAT  | THCn Histogram Frame Count Status |
| PVP_THCn_HCNT0_STAT  | THCn Histogram Counter Value 0    |
| PVP_THCn_HCNT1_STAT  | THCn Histogram Counter Value 1    |
| PVP_THCn_HCNT2_STAT  | THCn Histogram Counter Value 2    |
| PVP_THCn_HCNT3_STAT  | THCn Histogram Counter Value 3    |
| PVP_THCn_HCNT4_STAT  | THCn Histogram Counter Value 4    |
| PVP_THCn_HCNT5_STAT  | THCn Histogram Counter Value 5    |
| PVP_THCn_HCNT6_STAT  | THCn Histogram Counter Value 6    |
| PVP_THCn_HCNT7_STAT  | THCn Histogram Counter Value 7    |
| PVP_THCn_HCNT8_STAT  | THCn Histogram Counter Value 8    |
| PVP_THCn_HCNT9_STAT  | THCn Histogram Counter Value 9    |
| PVP_THCn_HCNT10_STAT | THCn Histogram Counter Value 10   |
| PVP_THCn_HCNT11_STAT | THCn Histogram Counter Value 11   |
| PVP_THCn_HCNT12_STAT | THCn Histogram Counter Value 12   |
| PVP_THCn_HCNT13_STAT | THCn Histogram Counter Value 13   |
| PVP_THCn_HCNT14_STAT | THCn Histogram Counter Value 14   |
| PVP_THCn_HCNT15_STAT | THCn Histogram Counter Value 15   |
| PVP_THCn_RREP_STAT   | THCn Number of RLE Reports        |

**Table 30-1:** PVP Module Registers (Continued)

| Name        | Description       |
|-------------|-------------------|
| PVP_PMA_CFG | PMA Configuration |

## ADSP-BF60x PVP Interrupt List

**Table 30-2:** ADSP-BF60x PVP Interrupt List

| Description                             | ID  | DMA Channel | Sensitivity |
|-----------------------------------------|-----|-------------|-------------|
| PVP0 Camera Pipe Data Out B DMA Channel | 111 | 38          | LEVEL       |
| PVP0 Camera Pipe Data Out C DMA Channel | 112 | 39          | LEVEL       |
| PVP0 Camera Pipe Status Out DMA Channel | 113 | 40          | LEVEL       |
| PVP0 Camera Pipe Control In DMA Channel | 114 | 41          | LEVEL       |
| PVP0 Status 0                           | 115 |             | LEVEL       |
| PVP0 Memory Pipe Data Out DMA Channel   | 116 | 42          | LEVEL       |
| PVP0 Memory Pipe Data In DMA Channel    | 117 | 43          | LEVEL       |
| PVP0 Memory Pipe Status Out DMA Channel | 118 | 44          | LEVEL       |
| PVP0 Memory Pipe Control In DMA Channel | 119 | 45          | LEVEL       |
| PVP0 Camera Pipe Data Out A DMA Channel | 120 | 46          | LEVEL       |
| PVP0 Status 1                           | 121 |             | LEVEL       |

## ADSP-BF60x PVP Trigger List

**Table 30-3:** Masters ADSP-BF60x PVP Trigger List

| Description                             | ID | Sensitivity |
|-----------------------------------------|----|-------------|
| PVP0 Camera Pipe Data Out B DMA Channel | 60 | PULSE/EDGE  |
| PVP0 Camera Pipe Data Out C DMA Channel | 61 | PULSE/EDGE  |
| PVP0 Camera Pipe Status Out DMA Channel | 62 | PULSE/EDGE  |
| PVP0 Camera Pipe Control In DMA Channel | 63 | PULSE/EDGE  |
| PVP0 Memory Pipe Data Out DMA Channel   | 64 | PULSE/EDGE  |
| PVP0 Memory Pipe Data In DMA Channel    | 65 | PULSE/EDGE  |
| PVP0 Memory Pipe Status Out DMA Channel | 66 | PULSE/EDGE  |
| PVP0 Memory Pipe Control In DMA Channel | 67 | PULSE/EDGE  |
| PVP0 Camera Pipe Data Out A DMA Channel | 68 | PULSE/EDGE  |
| PVP0 Status 0                           | 78 | LEVEL       |
| PVP0 Status 1                           | 79 | LEVEL       |
|                                         |    |             |

**Table 30-4:** Slaves ADSP-BF60x PVP Trigger List

| Description                             | ID | Sensitivity |
|-----------------------------------------|----|-------------|
| PVP0 Camera Pipe Data Out B DMA Channel | 60 |             |
| PVP0 Camera Pipe Data Out C DMA Channel | 61 |             |
| PVP0 Camera Pipe Status Out DMA Channel | 62 |             |
| PVP0 Camera Pipe Control In DMA Channel | 63 |             |
| PVP0 Memory Pipe Data Out DMA Channel   | 64 |             |
| PVP0 Memory Pipe Data In DMA Channel    | 65 |             |
| PVP0 Memory Pipe Status Out DMA Channel | 66 |             |
| PVP0 Memory Pipe Control In DMA Channel | 67 |             |
| PVP0 Camera Pipe Data Out A DMA Channel | 68 |             |
|                                         |    |             |

## PVP Block Diagram

The following figure shows the PVP block diagram. For more information on parts of this diagram, see the corresponding section of the [PVP Functional Description](#) or see [Architectural Concepts](#).



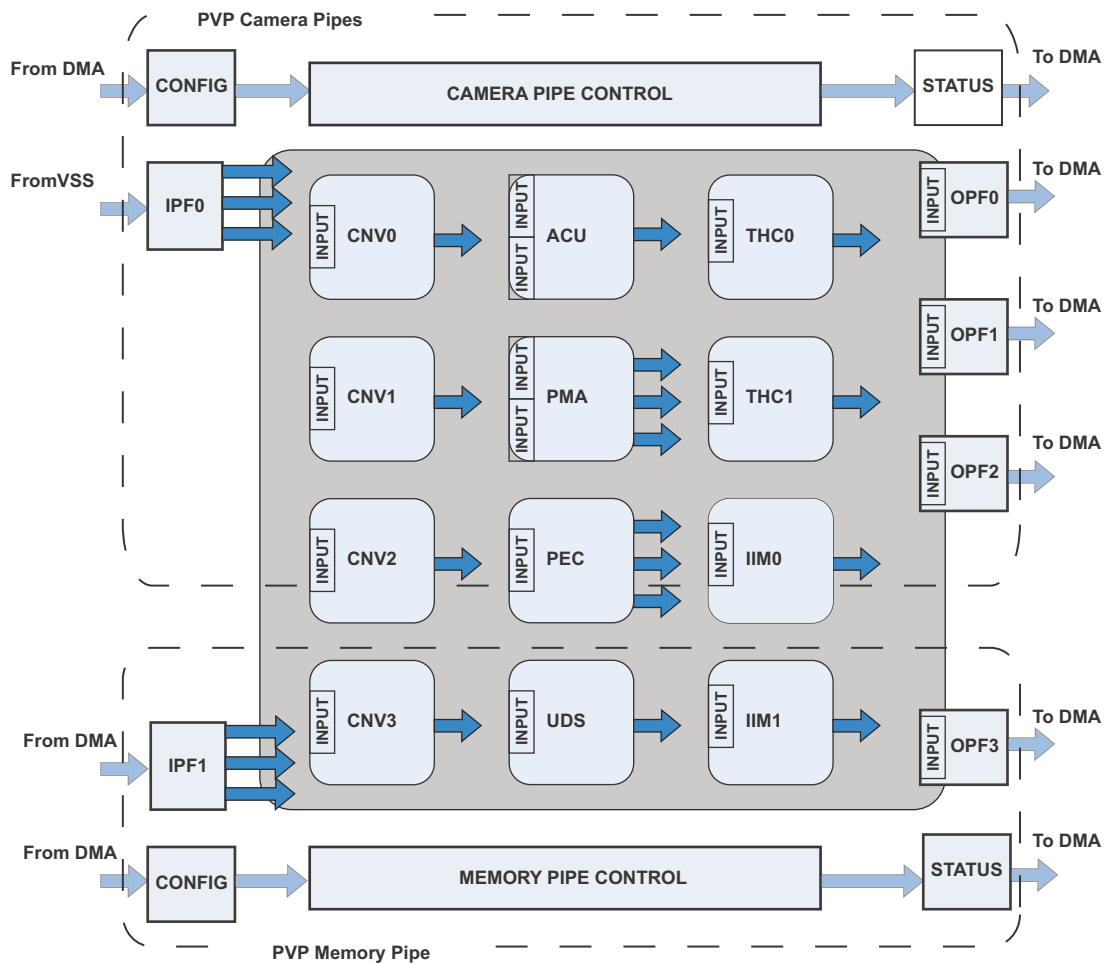


**Figure 30-1:** PVP High-Level Block Diagram

## Input Formatters (IPFx)

The PVP features two input formatters. The camera pipe is served by input formatter 0 (IPF0), where data is received directly from the video subsystem and the memory pipe is served by input formatter 1 (IPF1) where data is received from DMA. The input formatters master the entire pipe associated with them. Data received by IPF0 can be broadcast and is sent to the DMA system via output formatters OPF0, OPF1 and OPF2. Data received by IPF1 is sent to the DMA system via output formatter 3 (OPF3). The input formatters are responsible for the following tasks.

- Formatting input data
- Extracting color or luminance components
- Windowing

- Separating odd and even pixels
- Frame count
- Status report control

The following tables illustrate input formatter connections. The data source is fixed to video subsystem for IPF0 and to the data input DMA for IPF1. Note that s16 = 16-bit signed data and u16 = 16-bit unsigned data etc.

**Table 30-5:** IPF0 Block Connectivity

| Input 0            |      | Input 1 |      | Output 0   | Output 1   | Output 2 |
|--------------------|------|---------|------|------------|------------|----------|
| s16, u16, s32, u32 |      |         |      | s32        | s32        | s32      |
| Block              | Port | Block   | Port |            |            |          |
| VSS                | 0    | N/A     | N/A  | Full Frame | Odd/Window | Even     |

**Table 30-6:** IPF1 Block Connectivity

| Input 0            |      | Input 1 |      | Output 0   | Output 1 | Output 2 |
|--------------------|------|---------|------|------------|----------|----------|
| s16, u16, s32, u32 |      |         |      | s32        | s32      | s32      |
| Block              | Port | Block   | Port |            |          |          |
| DMA                | 0    | N/A     | N/A  | Full Frame | Odd      | Even     |

Both input formatters feature three 32-bit outputs that can be individually enabled by the respective `PVPO_IPFx_CTL.OPORTyEN` bits. All three outputs can operate in parallel. Output port 0 always outputs all data words, if enabled.

## Odd/Even Outputs

The IPF output ports 1 and 2 can partner in a way that all even-indexed data words output via port 2 and all odd-indexed data words output via port 1. In this mode, the even-indexed data words are delayed by one clock cycle where a pair of even/odd words leaves the IPFx simultaneously. The following figure illustrates the value of this feature:

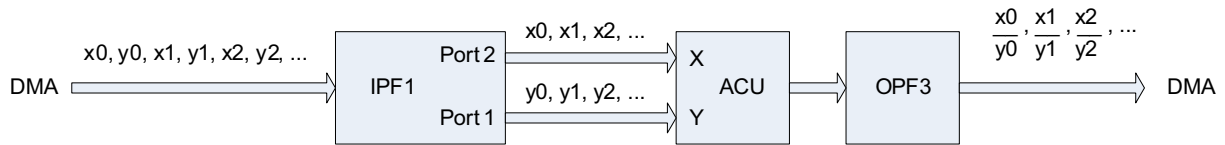


**Figure 30-2:** Separation of Odd and Even Data Words

For example, if the arithmetic control unit (ACU) is configured to perform division operations in memory pipe mode, it expects the numerator on its X input and the denominator on its Y input. The memory pipe receives all data in a single DMA channel (that may operate in 2-dimensional mode). The DMA may provide a series of interleaved numerator/denominator pairs. The IPF1 splits the data words as required. The resulting quotients are then sent to DMA by the output formatter OPF3.

## Windowing

Alternatively, in camera pipe mode the IPF0's output port 1 can output only the pixels which belong inside of the window coordinates as specified by the PVP\_IPF0\_HPOS, PVP\_IPF0\_VPOS, PVP\_IPF0\_HCNT and PVP\_IPF0\_VCNT registers.

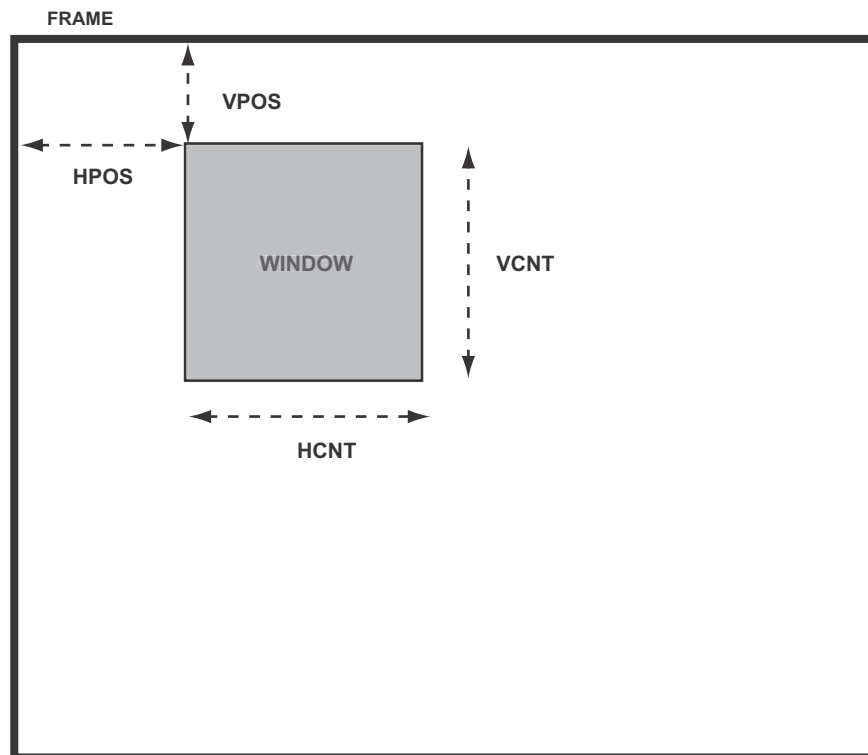


**Figure 30-3:** Definition of Port 1 Window Coordinates

The IPF1 (memory pipe) does not support windowing this way. Rather, the PVP\_IPF1\_HPOS and PVP\_IPF1\_VPOS registers are unused. The PVP\_IPF1\_HCNT and PVP\_IPF1\_VCNT registers are required to

match the `XCNT`, `YCNT` and `MSIZE` settings of the input data DMA. Window coordinates apply after color extraction, therefore consider the color format selected by the `PVP_IPF1_CTL.CFRMT` bit field.

## Monochrome Data Reception

Data is received from video subsystem (IPF0) or from data input DMA (IPF1) through 32-bit buses and IPF1 can receive 32-bit data words from the DMA. However, video pixels are typically represented by 16-bit data entities. For best system efficiency, pairs of 16-bit data words can be packed by the data source. The DMA controller enables 32-bit packing by the `DMAx_CFG.PSIZE = 2` bit setting. The parallel peripheral interfaces (PPIs) and the pixel compositor (PIXC) send packed color formats or the PPIs enable packing by the `PPIx_CTL.PACKEN` bit. Either way, both IPFs can unpack the incoming data to two 16-bit entities when the `PVP_IPFx_CTL.UNPACK` control bit is set.

The IPFs always output 32-bit words. By default, incoming data is zero extended to 32 bits. The `PVP_IPFx_CTL.SIGNEXT` bit changes the behavior to sign extension. The setting of the `PVP_IPFx_CTL.CFRMT` field determines how many significant bits the incoming data has. Monochrome data can have 8, 16, 24 or 32 significant bits. Note that, if the PPI receives 10-, 12- or 14-bit data it performs zero/sign-extension to 16 bits. In special color formats the IPF also extends from 5 or 6 significant bits.

Dimensions of a frame supplied to IPF0 and IPF1 generally must be a multiple of four. Further restrictions may apply by specific PVP blocks, especially if the odd/even ports are used or complex CFRMT formats are selected.

## Receiving Unsigned Data

Received video data is typically 8, 10, 12 or 14 bits wide and zero extended to 16-bit values. Therefore, pixels can always be positive values whether processed by signed or unsigned computation engines. If received data is however truly 16 or 32 bits wide it matters whether data is processed the signed or the unsigned way.

If the received data is 16- or 32-bit unsigned data and the IPFs forward the data to functional blocks that can only operate on 16-bit signed (CNVx, PMA) or 32-bit signed (ACU) data format, conflicts can occur. To avoid these conflicts, the `PVP_IPFx_CTL.QFRMT` bit instructs the IPF to right shift the incoming data by one bit position and to fill the most significant bit with a zero value. This way, the input is always interpreted as a positive value by signed engines. The output formatter (OPFx) blocks provide the counterpart functionality, which shifts the result back to the left by one bit position. This way, conflicts due to signed bits are avoided for the cost of losing one bit of input accuracy.

## Color Extraction

The PVP processes in monochrome data format. However, its data inputs often have to probe color data streams. Therefore, the IPFs feature a variety of features that allow extraction of certain luminance or chromatic values out of composite colored data streams. These features include:

- Extraction of Y (luma) values out of YCrCb streams
- Extraction of G (green) values out of RGB streams

- Extraction of G (green) values out of Bayer streams
- Extraction of R (red) values out of Bayer streams

The PVP\_IPFX\_CTL.CFRMT bit field defines what color format is used on the incoming data. The table “Supported Monochrome and Color Video Formats” provides the overview where each cell shows two consecutive 32-bit words that follow the conventions shown below. This refers to data sent by EPPI/PIXC in case of IPF0 and data stored in memory (and fetched by DMA) for IPF1.

|                  |                 |                 |                  |
|------------------|-----------------|-----------------|------------------|
| 1st word MS byte | 1st word byte 2 | 1st word byte 1 | 1st word LS byte |
| 2nd word MS byte | 2nd word byte 2 | 2nd word byte 1 | 2nd word LS byte |

For simplicity, YUV terminology is used rather than YCrCb. Items marked with \* indicate that multiple options are commonly used in industry standard, but there is no difference for the PVP as Cr and Cb components are discarded.

**Table 30-7:** Supported Monochrome and Color Video Formats

| CFRMT |             | Unpack = 0                                                                      | Unpack = 1 |    |    |    |                                                                               |     |    |    |    |
|-------|-------------|---------------------------------------------------------------------------------|------------|----|----|----|-------------------------------------------------------------------------------|-----|----|----|----|
| 0x1B  | 32-bit mono | <table><tr><td colspan="2">Y0</td></tr><tr><td colspan="2">Y1</td></tr></table> | Y0         |    | Y1 |    | <table><tr><td colspan="2">N/A</td></tr></table>                              | N/A |    |    |    |
| Y0    |             |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| Y1    |             |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| N/A   |             |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| 0x1A  | 24-bit mono | <table><tr><td>0</td><td>Y0</td></tr><tr><td>0</td><td>Y1</td></tr></table>     | 0          | Y0 | 0  | Y1 | <table><tr><td colspan="2">N/A</td></tr></table>                              | N/A |    |    |    |
| 0     | Y0          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| 0     | Y1          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| N/A   |             |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| 0x19  | 16-bit mono | <table><tr><td>0</td><td>Y0</td></tr><tr><td>0</td><td>Y1</td></tr></table>     | 0          | Y0 | 0  | Y1 | <table><tr><td>Y1</td><td>Y0</td></tr><tr><td>Y3</td><td>Y2</td></tr></table> | Y1  | Y0 | Y3 | Y2 |
| 0     | Y0          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| 0     | Y1          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| Y1    | Y0          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |
| Y3    | Y2          |                                                                                 |            |    |    |    |                                                                               |     |    |    |    |

**Table 30-7:** Supported Monochrome and Color Video Formats (Continued)

| CFRMT        |                | Unpack = 0                                                                                                          |    |  |  | Unpack = 1 |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
|--------------|----------------|---------------------------------------------------------------------------------------------------------------------|----|--|--|------------|-----------------------------------------------------------------------------------|---|----|-------------------------------------------------------------------------------|----|----|--------|---------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|----|----|----|----|----|----|
| 0x18         | 8-bit mono     | <table><tr><td></td><td></td><td></td><td>Y0</td></tr><tr><td></td><td></td><td></td><td>Y1</td></tr></table>       |    |  |  |            |                                                                                   |   | Y0 |                                                                               |    |    | Y1     | <table><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>Y7</td><td>Y6</td><td>Y5</td><td>Y4</td></tr></table> |    |    |    | Y3 | Y2 | Y1 | Y0 | Y7 | Y6 | Y5 | Y4 |
|              |                |                                                                                                                     | Y0 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
|              |                |                                                                                                                     | Y1 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y3           | Y2             | Y1                                                                                                                  | Y0 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y7           | Y6             | Y5                                                                                                                  | Y4 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0x14         | 16-bit YUV 422 | <table><tr><td>0</td><td>V0</td></tr><tr><td>0</td><td>Y1</td></tr></table>                                         |    |  |  | 0          | V0                                                                                | 0 | Y1 | <table><tr><td>Y0</td><td>V0</td></tr><tr><td>Y1</td><td>U0</td></tr></table> |    |    |        | Y0                                                                                                                        | V0 | Y1 | U0 |    |    |    |    |    |    |    |    |
| 0            | V0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0            | Y1             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y0           | V0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y1           | U0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0x15         | 16-bit YUV 422 | <table><tr><td>N/A</td></tr></table>                                                                                |    |  |  | N/A        | <table><tr><td>U0</td><td>V0</td></tr><tr><td>Y2</td><td>Y0</td></tr></table>     |   |    |                                                                               | U0 | V0 | Y2     | Y0                                                                                                                        |    |    |    |    |    |    |    |    |    |    |    |
| N/A          |                |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| U0           | V0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y2           | Y0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0x16         | 16-bit YUV 422 | <table><tr><td>N/A</td></tr></table>                                                                                |    |  |  | N/A        | <table><tr><td>Y1</td><td>Y0</td></tr><tr><td>U0/V1*</td><td>V0</td></tr></table> |   |    |                                                                               | Y1 | Y0 | U0/V1* | V0                                                                                                                        |    |    |    |    |    |    |    |    |    |    |    |
| N/A          |                |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y1           | Y0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| U0/V1*       | V0             |                                                                                                                     |    |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0x10<br>0x13 | 8-bit YUV 422  | <table><tr><td>0</td><td>0</td><td>0</td><td>V0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>Y0</td></tr></table> |    |  |  | 0          | 0                                                                                 | 0 | V0 | 0                                                                             | 0  | 0  | Y0     | <table><tr><td>Y1</td><td>U0</td><td>Y0</td><td>V0</td></tr><tr><td>Y3</td><td>U1</td><td>Y2</td><td>V1</td></tr></table> |    |    |    | Y1 | U0 | Y0 | V0 | Y3 | U1 | Y2 | V1 |
| 0            | 0              | 0                                                                                                                   | V0 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| 0            | 0              | 0                                                                                                                   | Y0 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y1           | U0             | Y0                                                                                                                  | V0 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |
| Y3           | U1             | Y2                                                                                                                  | V1 |  |  |            |                                                                                   |   |    |                                                                               |    |    |        |                                                                                                                           |    |    |    |    |    |    |    |    |    |    |    |

**Table 30-7:** Supported Monochrome and Color Video Formats (Continued)

| CFRMT |                     | Unpack = 0                                                                                                            | Unpack = 1                                                                                                                            |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
|-------|---------------------|-----------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|----|----|----|-------------------------------------------------------------------------------|-------|-------|-------|---------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|----|----|----|
| 0x11  | 8-bit<br>YUV<br>422 | <div>N/A</div>                                                                                                        | <table><tr><td>U1</td><td>V1</td><td>U0</td><td>V0</td></tr><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr></table>             | U1 | V1 | U0 | V0                                                                            | Y3    | Y2    | Y1    | Y0                                                                                                                        |    |    |    |    |    |    |    |    |
| U1    | V1                  | U0                                                                                                                    | V0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| Y3    | Y2                  | Y1                                                                                                                    | Y0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0x12  | 8-bit<br>YUV<br>422 | <div>N/A</div>                                                                                                        | <table><tr><td>Y3</td><td>Y2</td><td>Y1</td><td>Y0</td></tr><tr><td>U/V1*</td><td>V/U1*</td><td>U/V0*</td><td>V/U0*</td></tr></table> | Y3 | Y2 | Y1 | Y0                                                                            | U/V1* | V/U1* | U/V0* | V/U0*                                                                                                                     |    |    |    |    |    |    |    |    |
| Y3    | Y2                  | Y1                                                                                                                    | Y0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| U/V1* | V/U1*               | U/V0*                                                                                                                 | V/U0*                                                                                                                                 |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0x04  | 16-bit<br>RGB       | <table><tr><td>0</td><td>R0</td></tr><tr><td>0</td><td>G0</td></tr></table>                                           | 0                                                                                                                                     | R0 | 0  | G0 | <table><tr><td>G0</td><td>R0</td></tr><tr><td>R1</td><td>B0</td></tr></table> | G0    | R0    | R1    | B0                                                                                                                        |    |    |    |    |    |    |    |    |
| 0     | R0                  |                                                                                                                       |                                                                                                                                       |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0     | G0                  |                                                                                                                       |                                                                                                                                       |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| G0    | R0                  |                                                                                                                       |                                                                                                                                       |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| R1    | B0                  |                                                                                                                       |                                                                                                                                       |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0x00  | 8-bit<br>RGB        | <table><tr><td>0</td><td>0</td><td>0</td><td>R0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>G0</td></tr></table>   | 0                                                                                                                                     | 0  | 0  | R0 | 0                                                                             | 0     | 0     | G0    | <table><tr><td>R1</td><td>B0</td><td>G0</td><td>R0</td></tr><tr><td>G2</td><td>R2</td><td>B1</td><td>G1</td></tr></table> | R1 | B0 | G0 | R0 | G2 | R2 | B1 | G1 |
| 0     | 0                   | 0                                                                                                                     | R0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0     | 0                   | 0                                                                                                                     | G0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| R1    | B0                  | G0                                                                                                                    | R0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| G2    | R2                  | B1                                                                                                                    | G1                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| 0x01  | 8-bit<br>RGB        | <table><tr><td></td><td>B0</td><td>G0</td><td>R0</td></tr><tr><td></td><td>B1</td><td>G1</td><td>R1</td></tr></table> |                                                                                                                                       | B0 | G0 | R0 |                                                                               | B1    | G1    | R1    | <table><tr><td>R1</td><td>B0</td><td>G0</td><td>R0</td></tr><tr><td>G2</td><td>R2</td><td>B1</td><td>G1</td></tr></table> | R1 | B0 | G0 | R0 | G2 | R2 | B1 | G1 |
|       | B0                  | G0                                                                                                                    | R0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
|       | B1                  | G1                                                                                                                    | R1                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| R1    | B0                  | G0                                                                                                                    | R0                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |
| G2    | R2                  | B1                                                                                                                    | G1                                                                                                                                    |    |    |    |                                                                               |       |       |       |                                                                                                                           |    |    |    |    |    |    |    |    |

**Table 30-7:** Supported Monochrome and Color Video Formats (Continued)

| CFRMT |                      | Unpack = 0                                                                      | Unpack = 1 |      |   |      |                                                                                       |      |      |      |      |
|-------|----------------------|---------------------------------------------------------------------------------|------------|------|---|------|---------------------------------------------------------------------------------------|------|------|------|------|
| 0x02  | 16-bit<br>RGB<br>565 | <table><tr><td>0</td><td>RGB0</td></tr><tr><td>0</td><td>RGB1</td></tr></table> | 0          | RGB0 | 0 | RGB1 | <table><tr><td>RGB1</td><td>RGB0</td></tr><tr><td>RGB3</td><td>RGB2</td></tr></table> | RGB1 | RGB0 | RGB3 | RGB2 |
| 0     | RGB0                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| 0     | RGB1                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| RGB1  | RGB0                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| RGB3  | RGB2                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| 0x03  | 24-bit<br>RGB<br>666 | <table><tr><td></td><td>RGB0</td></tr><tr><td></td><td>RGB1</td></tr></table>   |            | RGB0 |   | RGB1 | <table><tr><td>RGB1</td><td>RGB0</td></tr><tr><td>RGB2</td><td>RGB1</td></tr></table> | RGB1 | RGB0 | RGB2 | RGB1 |
|       | RGB0                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
|       | RGB1                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| RGB1  | RGB0                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |
| RGB2  | RGB1                 |                                                                                 |            |      |   |      |                                                                                       |      |      |      |      |

The IPFs process all data words for the monochrome input format. For YCrCb formats, the IPFs discard the Cr and Cb color components and only process the luminance values Y. With RGB formats the PVP\_IPFx\_CTL.EXTRED bit is used. If cleared, the IPFs extract the green components. If set, the IPFs extract the red components.

On top of the streaming formats shown in the table above, the IPFs can also extract color components out of Bayer pattern matrices as shown in the figure below.

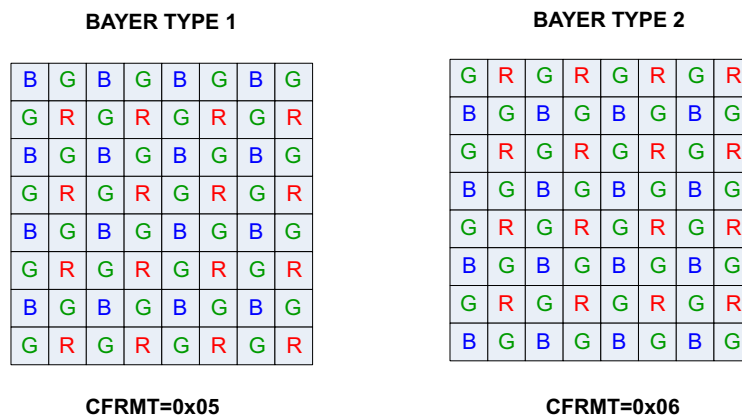


**Figure 30-4:** Supported Bayer Patterns

If the PVP\_IPFx\_CTL.EXTRED bit =0, the IPFs extract the green components of the Bayer matrix. The output frame has half the horizontal resolution of the input frame. If the PVP\_IPFx\_CTL.EXTRED bit =1, the IPFs extract the red component, dividing horizontal and vertical resolution.

In Bayer extraction mode, the input values can be 10, 12 or 16 bits wide. An 8-bit format is not supported. The PPI's SWAPEN bit must be cleared in this mode.



The Bayer concepts can also apply to red-clear-clear-clear (RCCC) data. This is similar to Bayer format, except that the green and blue pixels have monochrome (clear filter) values. Programs have then the following options:

- Extract the red pixels as in Bayer extraction mode.
- Extract one clear pixel by swapping type 1 vs. type 2 and setting the EXTRED bit.
- Extract the diagonal clear pixels that correspond to the green pixels in Bayer format.
- Use the odd/even mechanism and only process the clear columns.
- Ignore the fact that red pixels are special and low-pass them using convolution blocks.

## Pipe Mastering

Beside formatting and broadcasting incoming data, the IPFs master the camera and memory pipelines. For a description of that functionality, see the Programming Model section.

## PPI and PVP Interoperability

If IPF0 is receiving data from any of the PPIs in camera pipe mode, the settings of the two modules need to partner for reception of reasonable data formats. The following Table lists the supported combinations of settings.

**Table 30-8:** Matching PPI and PVP Settings

| Pixel<br>Format<br>on PPI<br>pins | PPI settings |          |                                                |                                                                    |                                 |                                 |                                                          | PVP Settings |                      |                       |                                    |
|-----------------------------------|--------------|----------|------------------------------------------------|--------------------------------------------------------------------|---------------------------------|---------------------------------|----------------------------------------------------------|--------------|----------------------|-----------------------|------------------------------------|
|                                   | XFR_<br>TYPE | DLE<br>N | S<br>P<br>L<br>I<br>T<br>_<br>W<br>O<br>R<br>D | S<br>P<br>L<br>I<br>T<br>_<br>E<br>V<br>E<br>N<br>_<br>O<br>D<br>D | P<br>A<br>C<br>K<br>E<br>D<br>N | D<br>M<br>A<br>_<br>C<br>F<br>G | S<br>U<br>B<br>S<br>P<br>L<br>I<br>T<br>_<br>O<br>D<br>D | CFORM<br>AT  | UN<br>PAC<br>KE<br>N | CFORMAT<br>Definition | Comments                           |
| RGB888<br>(8<br>bits/pix<br>clk)  | GP           | 8        | 0                                              | 0                                                                  | 0                               | X                               | X                                                        | 00000        | 0                    | RGB 8 bit             | RGBRGB... but not packed<br>on PxP |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings                            |              |                  |                                                |                                                                    |                            |                                 |                                                          | PVP Settings |                      |                       |                                    |
|-----------------------------------------|--------------|------------------|------------------------------------------------|--------------------------------------------------------------------|----------------------------|---------------------------------|----------------------------------------------------------|--------------|----------------------|-----------------------|------------------------------------|
| Pixel<br>Format<br>on PPI<br>pins       | XFR_<br>TYPE | DLE<br>N         | S<br>P<br>L<br>I<br>T<br>_<br>W<br>O<br>R<br>D | S<br>P<br>L<br>I<br>T<br>_<br>E<br>V<br>E<br>N<br>_<br>O<br>D<br>D | P<br>A<br>C<br>K<br>E<br>N | D<br>M<br>A<br>_<br>C<br>F<br>G | S<br>U<br>B<br>S<br>P<br>L<br>I<br>T<br>_<br>O<br>D<br>D | CFORM<br>AT  | UN<br>PAC<br>KE<br>N | CFORMAT<br>Definition | Comments                           |
| RGB888<br>(8<br>bits/pix<br>clk)        | GP           | 8                | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00000        | 1                    | RGB 8 bit             | RGBRGB... but packed on<br>PxP     |
| RGB888<br>(24<br>bits/pix<br>clk)       | GP           | 24               | 0                                              | 0                                                                  | 0                          | X                               | X                                                        | 00001        | 0                    | RGB 888               | RGB...RGB , not packed on<br>PxP   |
| RGB888<br>(24<br>bits/pix<br>clk)       | GP           | 24               | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00001        | 1                    | RGB 888               | RGB...RGB , packed on PxP          |
| RGB(10<br>/12/16<br>bits per<br>pixclk) | GP           | 10/<br>12/<br>16 | 0                                              | 0                                                                  | 0                          | X                               | X                                                        | 00100        | 0                    | RGB 16 bit            | RGBRGB... but not packed<br>on PxP |
| RGB(10<br>/12/16<br>bits per<br>pixclk) | GP           | 10/<br>12/<br>16 | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00100        | 0                    | RGB 16 bit            | RGBRGB... but packed on<br>PxP     |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings                      |              |                  |                                                |                                                                    |                            |                                 |                                                          | PVP Settings |                      |                       |                                      |
|-----------------------------------|--------------|------------------|------------------------------------------------|--------------------------------------------------------------------|----------------------------|---------------------------------|----------------------------------------------------------|--------------|----------------------|-----------------------|--------------------------------------|
| Pixel<br>Format<br>on PPI<br>pins | XFR_<br>TYPE | DLE<br>N         | S<br>P<br>L<br>I<br>T<br>_<br>W<br>O<br>R<br>D | S<br>P<br>L<br>I<br>T<br>_<br>E<br>V<br>E<br>N<br>_<br>O<br>D<br>D | P<br>A<br>C<br>K<br>E<br>N | D<br>M<br>A<br>_<br>C<br>F<br>G | S<br>U<br>B<br>S<br>P<br>L<br>I<br>T<br>_<br>O<br>D<br>D | CFORM<br>AT  | UN<br>PAC<br>KE<br>N | CFORMAT<br>Definition | Comments                             |
| RGB565<br>(16<br>bits/pix<br>clk) | GP           | 16               | 0                                              | 0                                                                  | 0                          | X                               | X                                                        | 00010        | 0                    | RGB 565 bit           | RGB...RGB , but not packed<br>on PxP |
| RGB565<br>(16<br>bits/pix<br>clk) | GP           | 16               | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00010        | 1                    | RGB 565 bit           | RGB...RGB , but packed on<br>PxP     |
| RGB666<br>(18<br>bits/pix<br>clk) | GP           | 18               | 0                                              | 0                                                                  | 0                          | X                               | X                                                        | 00011        | 0                    | RGB 666 bit           | RGB...RGB , but not packed<br>on PxP |
| RGB666<br>(18<br>bits/pix<br>clk) | GP           | 18               | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00011        | 1                    | RGB 666 bit           | RGB...RGB , but packed on<br>PxP     |
| Bayer<br>format<br>type-1         | GP           | 10/<br>12/<br>16 | 0                                              | 0                                                                  | 1                          | X                               | X                                                        | 00101        | 1                    | Bayer type-1          |                                      |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings              |            |          |            |           |       |         |             | PVP Settings |          |                            |                                      |
|---------------------------|------------|----------|------------|-----------|-------|---------|-------------|--------------|----------|----------------------------|--------------------------------------|
| Pixel Format on PPI pins  | XFR_TYPE   | DLEN     | SPLIT_WORD | SPLIT_VEN | PACKE | DMA_CFG | SUBSPLIT_OD | CFORMAT      | UNPACKEN | CFORMAT Definition         | Comments                             |
| Bayer type-2              | GP         | 10/12/16 | 0          | 0         | 1     | X       | X           | 00110        | 1        | Bayer type-2               |                                      |
| CbYCrY(8 bits/pix clk)    | ITU-656/GP | 8        | 0          | 0         | 0     | X       | X           | 10000        | 0        | YUV 422 8bit               | VYUY... but not packed on PxP        |
| CbYCrY(8 bits/pix clk)    | ITU-656/GP | 8        | 0          | 0         | 1     | X       | X           | 10000        | 1        | YUV 422 8bit               | VYUY... but packed on PxP            |
| CbY..CrY(16 bits/pix clk) | GP         | 16       | 0          | 0         | 1     | 0       | 0           | 10011        | 1        | YUV 422 8bit pair in 16bit | VYUY... but packed on PxP            |
| YYYY(8 bits/pix clk)      | ITU-656/GP | 8        | 0          | 0         | 0     | X       | X           | 11000        | 0        | Y alone 8 bit              | YYYY... , not packed , All Ys on PxP |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings             |            |      |            |           |        |         |          | PVP Settings |          |                            |                                                                            |
|--------------------------|------------|------|------------|-----------|--------|---------|----------|--------------|----------|----------------------------|----------------------------------------------------------------------------|
| Pixel Format on PPI pins | XFR_TYPE   | DLEN | SPLIT_WORD | SPLIT_VEN | PACKEN | DMA_CFG | SUBSPLIT | CFORMAT      | UNPACKEN | CFORMAT Definition         | Comments                                                                   |
| YYYY(8 bits/pix clk)     | ITU-656/GP | 8    | 0          | 0         | 1      | X       | X        | 11000        | 1        | Y alone 8 bit              | YYYY... , packed on PxP , All Ys on PxP                                    |
| CbYCrY(8 bits/pix clk)   | ITU-656/GP | 8    | 0          | 1         | X      | 1       | X        | 11000        | 1        | Y alone 8 bit              | VYUY... , split and DMA separated , All Ys on PxP                          |
| CbYCrY(8 bits/pix clk)   | ITU-656/GP | 8    | 0          | 1         | X      | 0       | 0        | 10001        | 1        | YUV 422 8bit split Y-next  | VYUY... , split , UV comes first , followed by Ys on PxP                   |
| CbYCrY(8 bits/pix clk)   | ITU-656/GP | 8    | 0          | 1         | X      | 0       | 1        | 10010        | 1        | YUV 422 8bit split Y-first | VYUY... , split & subsplit , Ys comes first , followed by Vs and Us on PxP |
| SMPTE (16 bits/pix clk)  | ITU-656/GP | 16   | 1          | 1         | X      | 0       | 1        | 10010        | 1        | YUV 422 8bit split Y-first | VY..UY , split & subsplit , Ys comes first , followed by Vs and Us on PxP  |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings                         |                    |                         |                                                |                                                               |                            |                                 |                                                     | PVP Settings |                      |                                  |                                                                                         |
|--------------------------------------|--------------------|-------------------------|------------------------------------------------|---------------------------------------------------------------|----------------------------|---------------------------------|-----------------------------------------------------|--------------|----------------------|----------------------------------|-----------------------------------------------------------------------------------------|
| Pixel<br>Format<br>on PPI<br>pins    | XFR_<br>TYPE       | DLE<br>N                | S<br>P<br>L<br>I<br>T<br>_<br>W<br>O<br>R<br>D | S<br>P<br>L<br>I<br>T<br>_<br>E<br>V<br>E<br>N<br>_<br>O<br>D | P<br>A<br>C<br>K<br>E<br>N | D<br>M<br>A<br>_<br>C<br>F<br>G | S<br>U<br>B<br>S<br>P<br>L<br>I<br>T<br>_<br>O<br>D | CFORM<br>AT  | UN<br>PAC<br>KE<br>N | CFORMAT<br>Definition            | Comments                                                                                |
| SMPTE<br>(16<br>bits/pix<br>clk)     | ITU-<br>656/G<br>P | 16                      | 1                                              | 1                                                             | X                          | 0                               | 0                                                   | 10010        | 1                    | YUV 422 8bit<br>split Y-first    | VY..UY , split , Ys comes<br>first , followed by VUs on<br>PxP                          |
| CbYCr<br>Y(10-16<br>bits/pix<br>clk) | ITU-<br>656/G<br>P | 10/<br>12/<br>14/<br>16 | 0                                              | 0                                                             | 0                          | X                               | X                                                   | 10100        | 0                    | YUV 422 16<br>bit                | VYUY... sign/zero ext to 16<br>bits, but not packed on PxP                              |
| CbYCr<br>Y(10-16<br>bits/pix<br>clk) | ITU-<br>656/G<br>P | 10/<br>12/<br>14/<br>16 | 0                                              | 0                                                             | 1                          | X                               | X                                                   | 10100        | 0                    | YUV 422 16<br>bit                | VYUY... sign/zero ext to 16<br>bits, but packed on PxP                                  |
| CbYCr<br>Y(10-16<br>bits/pix<br>clk) | ITU-<br>656/G<br>P | 10/<br>12/<br>14/<br>16 | 0                                              | 1                                                             | X                          | 0                               | 0                                                   | 10101        | 1                    | YUV 422<br>16bit split<br>Y-next | VYUY... ,sign/zero ext to 16<br>bits, split , UV comes first ,<br>followed by Ys on PxP |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings               |            |             |            |           |        |         |          | PVP Settings |          |                             |                                                                                                     |
|----------------------------|------------|-------------|------------|-----------|--------|---------|----------|--------------|----------|-----------------------------|-----------------------------------------------------------------------------------------------------|
| Pixel Format on PPI pins   | XFR_TYPE   | DLEN        | SPLIT_WORD | SPLIT_VEN | PACKEN | DMA_CFG | SUBSPLIT | CFORMAT      | UNPACKEN | CFORMAT Definition          | Comments                                                                                            |
| CbYCrY(10-16 bits/pix clk) | ITU-656/GP | 10/12/14/16 | 0          | 1         | X      | 0       | 1        | 10110        | 1        | YUV 422 16bit split Y-first | VY..UY , sign/zero ext to 16 bits, split & subsplit , Ys comes first , followed by Vs and Us on PxP |
| SMPTE (20/24 bits/pix clk) | ITU-656/GP | 20/24       | 1          | 1         | X      | 0       | 1        | 10110        | 1        | YUV 422 16bit split Y-first | VY..UY , sign/zero ext to 16 bits, split & subsplit , Ys comes first , followed by Vs and Us on PxP |
| SMPTE (20/24 bits/pix clk) | ITU-656/GP | 20/24       | 1          | 1         | X      | 0       | 0        | 10110        | 1        | YUV 422 16bit split Y-first | VY..UY , sign/zero ext to 16 bits, split , Ys comes first , followed by VUs on PxP                  |
| YYYY(10-16 bits/pix clk)   | ITU-656/GP | 10/12/14/16 | 0          | 0         | 0      | X       | X        | 11001        | 0        | Y alone 16 bit              | YYYY... , sign/zero ext to 16 bits, not packed , All Ys on PxP                                      |
| YYYY(10-16 bits/pix clk)   | ITU-656/GP | 10/12/14/16 | 0          | 0         | 1      | X       | X        | 11001        | 1        | Y alone 16 bit              | YYYY... , sign/zero ext to 16 bits, packed , All Ys on PxP                                          |
| SMPTE (20/24 bits/pix clk) | ITU-656/GP | 20/24       | 1          | 1         | X      | 1       | X        | 11001        | 1        | Y alone 16 bit              | VY..UY... , sign/zero ext to 16 bits, split and DMA separated , All Ys on PxP                       |

**Table 30-8:** Matching PPI and PVP Settings (Continued)

| PPI settings             |          |      |             |                |        |              |              | PVP Settings |          |                    |                                         |
|--------------------------|----------|------|-------------|----------------|--------|--------------|--------------|--------------|----------|--------------------|-----------------------------------------|
| Pixel Format on PPI pins | XFR_TYPE | DLEN | SPLIT_WORDS | SPLIT_ELEMENTS | PACKED | DMA_CHANNELS | SUBSPLIT_ODD | CFORMAT      | UNPACKEN | CFORMAT Definition | Comments                                |
| YYYY(24 bits/pixel)      | GP       | 24   | 0           | 0              | 0      | X            | X            | 11010        | 0        | Y alone 24 bit     | YYYY... , not packed , All Ys on PxP    |
| YYYY(32 bits/pixel)      | GP       | 32   | 0           | 0              | 0      | X            | X            | 11011        | 0        | 32 bit             | YYYY... , packed on PxP , All Ys on PxP |

## Output Formatters (OPFx)

The output formatters collect the data results of PVP processing blocks, apply final formatting and then forward the results to the DMA channels. The OPF0, OPF1 and OPF2 formatters serve the three camera pipes, while OPF3 serves the memory pipe. Each OPF is associated with a specific DMA channel. The single input is widely configurable as shown in the following tables.

**Table 30-9:** OPF0, OPF1, OPF2 Block Connectivity

| Input 0            | Input 1 | Output 0           | Output 1 | Output 2 |
|--------------------|---------|--------------------|----------|----------|
| s16, u16, s32, u32 |         | s16, u16, s32, u32 |          |          |



**Table 30-9:** OPF0, OPF1, OPF2 Block Connectivity (Continued)

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| Block   | Port    | Block   | Port |          |          |          |
| IPF1    | 0, 1, 2 | n/a     | n/a  | DMA      | n/a      | n/a      |
| CNV0    | 0       |         |      |          |          |          |
| CNV1    | 0       |         |      |          |          |          |
| CNV2    | 0       |         |      |          |          |          |
| CNV3    | 0       |         |      |          |          |          |
| PMA     | 0, 1, 2 |         |      |          |          |          |
| ACU     | 0       |         |      |          |          |          |
| PEC     | 0       |         |      |          |          |          |
| THC0    | 0       |         |      |          |          |          |
| THC1    | 0       |         |      |          |          |          |
| IIM0    | 0       |         |      |          |          |          |
| IIM1    | 0       |         |      |          |          |          |

**Table 30-10:** OPF3 Block Connectivity

| Input 0               |         | Input 1 |      | Output 0              | Output 1 | Output 2 |
|-----------------------|---------|---------|------|-----------------------|----------|----------|
| s16, u16,<br>s32, u32 |         |         |      | s16, u16, s32,<br>u32 |          |          |
| Block                 | Port    | Block   | Port |                       |          |          |
| IPF1                  | 0, 1, 2 | n/a     | n/a  | DMA                   | n/a      | n/a      |
| CNV0                  | 0       |         |      |                       |          |          |
| CNV1                  | 0       |         |      |                       |          |          |
| CNV2                  | 0       |         |      |                       |          |          |
| CNV3                  | 0       |         |      |                       |          |          |
| PMA                   | 0, 1, 2 |         |      |                       |          |          |
| ACU                   | 0       |         |      |                       |          |          |
| PEC                   | 0       |         |      |                       |          |          |
| THC0                  | 0       |         |      |                       |          |          |
| THC1                  | 0       |         |      |                       |          |          |
| IIM0                  | 0       |         |      |                       |          |          |
| IIM1                  | 0       |         |      |                       |          |          |
| UDS                   | 0       |         |      |                       |          |          |

## Data Packing

To ensure efficient memory bandwidth use, the OPFs support data packing. Ideally, the PVP results are packed into 32-bit entities and forwarded to the DMA channel, which is also programmed to 32-bit mode by setting `PVP_OPFx_CTL.OSIZE = 0` and the `DMA_CFG.PSIZE = 2` for each respective DMA channel.

The `PVP_OPFx_CTL.ISIZE = 0` setting instructs the OPF to accept a full 32-bit word from the data source. In this case, packing is not possible and `PVP_OPFx_CTL.OSIZE = 0` and `DMA_CFG.PSIZE = 2` are mandatory settings.

The `PVP_OPFx_CTL.ISIZE = 1` setting instructs the OPF to accept 16-bit words from the data source. In this case the user has multiple options. If packing is desired, set `PVP_OPFx_CTL.OSIZE = 0` and `DMA_CFG.PSIZE = 2`. Otherwise, these bit fields are set to `PVP_OPFx_CTL.OSIZE = 1` and `DMA_CFG.PSIZE = 1` for 16-bit transfers between OPF and DMA channel. The `PVP_OPFx_CTL.IUP16` bit determines whether the OPFs latch the lower 16 bits or the upper 16 bits from their 32-bit input.

When `PVP_OPFx_CTL.ISIZE = 2` the OPF to accepts 8-bit bytes from the data source. To pack values to a 32-bit output, set `PVP_OPFx_CTL.OSIZE = 0` and `DMA_CFG.PSIZE = 2`. To only pack two values into a 16-bit output, set `PVP_OPFx_CTL.OSIZE = 1` and `DMA_CFG.PSIZE = 1`. To disable packing, set `PVP_OPFx_CTL.OSIZE = 2` and `DMA_CFG.PSIZE = 0`. Again, the `PVP_OPFx_CTL.IUP16` bit selects between bits [7:0] and bits [23:16] on the inputs.

The `PVP_OPFx_CTL.ISIZE = 3` setting instructs the OPF to accept 4-bit nibbles from the data source. Nibbles must always be packed to 8-, 16- or 32-bit entities following the rules already described above. This results in dual, quad or octal nibble groups per DMA transfer. The `PVP_OPFx_CTL.IUP16` bit selects between bits [3:0] and bits [19:16] on the inputs.

Note that data packing has requirements for the horizontal size of data frames, which must be a multiple of the packing ratio. The following table summarizes the packing options.

**Table 30-11:** OPF Packing Options

| ISIZE | OSIZE | PSIZE | IUP16 | 32-bit DMA word         |
|-------|-------|-------|-------|-------------------------|
| 0     | 0     | 2     | 0     | D0[31:0]                |
| 1     | 1     | 1     | 0     | 0, D0[15:0]             |
|       |       |       | 1     | 0, D0[31:16]            |
|       | 0     | 2     | 0     | D1[15:0], D0[15:0]      |
|       |       |       | 1     | D1[31:16],<br>D0[31:16] |

**Table 30-11:** OPF Packing Options (Continued)

| ISIZE | OSIZE | PSIZE | IUP16 | 32-bit DMA word                                     |
|-------|-------|-------|-------|-----------------------------------------------------|
| 2     | 2     | 0     | 0     | 0, 0, 0, D0[7:0]                                    |
|       |       |       | 1     | 0, 0, 0, D0[23:16]                                  |
|       | 1     | 1     | 0     | 0, 0, D1[7:0],<br>D0[7:0]                           |
|       |       |       | 1     | 0, 0, D1[23:16],<br>D0[23:16]                       |
|       | 0     | 2     | 0     | D3[7:0], D2[7:0],<br>D1[7:0], D0[7:0]               |
|       |       |       | 1     | D3[23:16],<br>D2[23:16],<br>D1[23:16],<br>D0[23:16] |

**Table 30-11:** OPF Packing Options (Continued)

| ISIZE | OSIZE | PSIZE | IUP16 | 32-bit DMA word                                                                                             |
|-------|-------|-------|-------|-------------------------------------------------------------------------------------------------------------|
| 3     | 2     | 0     | 0     | 0, 0, 0, 0, 0, 0,<br>D1[3:0], D0[3:0]                                                                       |
|       |       |       | 1     | 0, 0, 0, 0, 0, 0,<br>D1[19:16],<br>D0[19:16]                                                                |
|       | 1     | 1     | 0     | 0, 0, 0, 0, D3[3:0],<br>D2[3:0], D1[3:0],<br>D0[3:0]                                                        |
|       |       |       | 1     | 0, 0, 0, 0, D3[19:16],<br>D2[19:16],<br>D1[19:16],<br>D0[19:16]                                             |
|       | 0     | 2     | 0     | D7[3:0], D6[3:0],<br>D5[3:0], D4[3:0],<br>D3[3:0], D2[3:0],<br>D1[3:0], D0[3:0]                             |
|       |       |       | 1     | D7[19:16],<br>D6[19:16],<br>D5[19:16],<br>D4[19:16],<br>D3[19:16],<br>D2[19:16],<br>D1[19:16],<br>D0[19:16] |

If the IPF is receiving unsigned data and its `PVP_IPFx_CTL.QFRMT` bit is set, it is helpful that the OPF corrects the shifting position back before results are transmit to DMA. For this purpose, the `PVP_OPFx_CTL.QFRMT` bit shifts the results to the left by one bit position.

## Output FIFOs

To avoid data loss in cases of temporary peak loads on system buses, the DMA FIFOs OPFs have significant local FIFOs. The width of the FIFO varies with the `PVP_OPFx_CTL.OSIZE` bit field settings. The individual OPFs have different FIFO depths:

- OPF0: 380 entries x 32 bits or 960 x 16 bits or 1920 x 8 bits
- OPF1: 1024 entries x 32 bits or 2048 x 16 bits or 4196 x 8 bits

- OPF2: 512 entries x 32 bits or 1024 x16 bits or 2048 x 8 bits
- OPF3: 32 entries x 32 bits or 64 x 16 bits or 128 x 8 bits

Therefore, it is important to assign OPF1 to the camera pipeline with the highest output data rate. For example, 32-bit results that come out of the CNVx, ACU or IIMx blocks, running at full input frame resolution, are better routed to OPF1. Output formatter 0 should be used for the camera pipe with the least output data rate. Typical candidates are items that are subject to data reduction (windowing, sub sampling, run-length encoding) or items that have small data widths (threshold index, edge classes). Output formatter 2 is designed for normal data rates. Unexpected overflows are reported by the OPF00VF, OPF10VF and OPF10VF status bits in the PVP\_STAT and PV00\_ILAT registers.

Output formatter 3 has a relatively small FIFO because the memory pipe has been designed to never overflow.

OPFs support the finish signaling to the DMA. This is described in detail later in the section [Finish Commands](#).

## Threshold-Histogram-Compression (THCx)

The PVP features two threshold-histogram-compression blocks. These blocks implement a collection of statistical and range reduction signal processing functions. The input to the THC blocks can be one of the following described in the following table and Figure 5 gives the detailed overview of the THC block

**Table 30-12:** THCx Block Connectivity

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| s32     |         |         |      | s32      |          |          |
| Block   | Port    | Block   | Port |          |          |          |
| IPF0    | 0,1, 2  | n/a     | n/a  | result   | n/a      | n/a      |
| IPF1    | 0, 1, 2 |         |      |          |          |          |
| PMA     | 1, 2    |         |      |          |          |          |
| CNV0    | 0       |         |      |          |          |          |
| CNV1    | 0       |         |      |          |          |          |
| CNV2    | 0       |         |      |          |          |          |
| CNV3    | 0       |         |      |          |          |          |
| ACU     | 0       |         |      |          |          |          |
| PEC     | 2       |         |      |          |          |          |

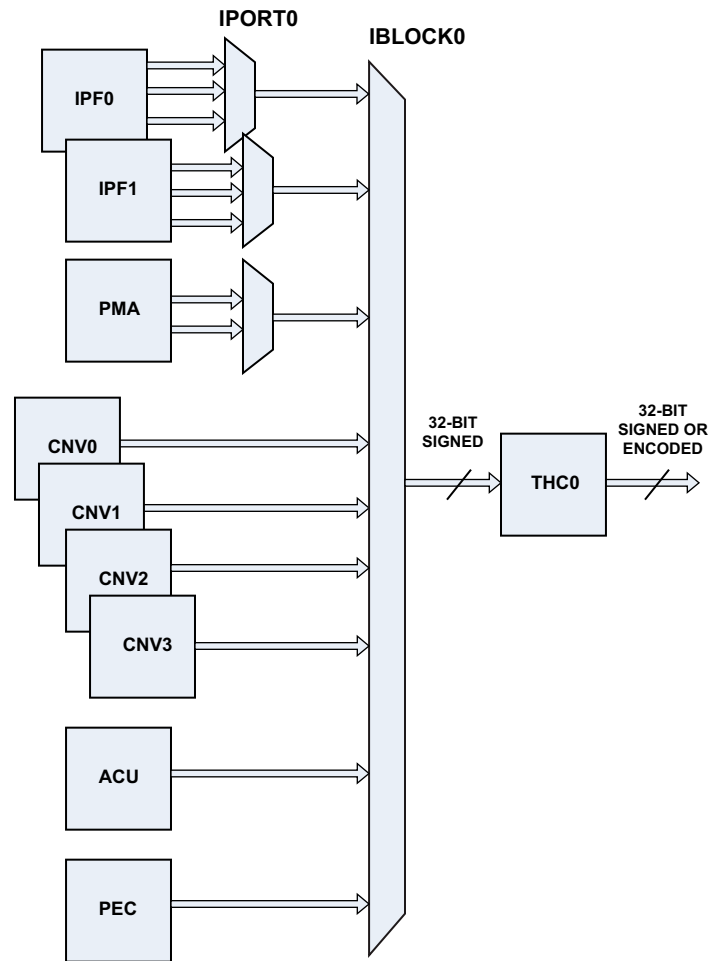


**Figure 30-5:** Threshold Histogram Compression Block Diagram

The `PVP_THCx_CTL.ZEXT` bit determines how the input data is handled. When set to 1, the lower 16-bits of input data are zero extended to 32-bits and used for further processing. The upper 16-bits are used as an angle input. When the bit is cleared ( $= 0$ ), the entire 32-bits of input data are used for further processing. The angle input is set to 0.

The three stages of THC block operation are as follows.

- Threshold Unit
- Histogram Unit
- Compression Unit

The details of each unit are described in the sections that follow.

The threshold unit operates on the incoming pixels for the THC block. The output of the threshold unit is a 32-bit pixel out value and a 4-bit index value.

- **Clipping Mode** — In this mode the input pixel is classified in one of the three ranges. The input data is first compared to the value in the PVP\_THCx\_CMINTH register. If this value less than the value programmed in the register then the pixel out value is the PVP\_THCx\_CMINVAL register value and the index output is 0x0. If the input data is greater than the value programmed in the PVP\_THCx\_CMAXTH register, then the pixel out value is the PVP\_THCx\_CMAXVAL value and the index output is 0x2. If the input data lies between the two values the pixel out value is equal to the input data and the index output is 0x1.
- **Quantization Mode** — In this mode the input pixels are compared to the 16 PVP\_THCx\_THy registers and classified as belonging to one of sixteen bins. The registers must be programmed in ascending order from PVP\_THCx\_TH0 to PVP\_THCx\_TH15. The pixel out takes one of these 16 threshold values

based on floor logic. If the input is less than PVP\_THCx\_TH0, then the pixel out is the highest threshold value among the 16 threshold values. The index output is the number of the threshold register that the pixel out takes.

- **Hysteresis Mode** — In this mode the 16 PVP\_THCx\_THy registers are interpreted as eight pairs of threshold values starting from (TH0–TH1), (TH2–TH3), and so on. The register pairs form a range. If the input pixel value falls in this range then the pixel out depends on the previous output. The pixel out is the higher register in one of the eight pairs of threshold registers. The index output is the number of the threshold register pair that the pixel out takes.

The next figure shows example hysteresis functionality. The rounded numbers are index output when the input pixel falls in that region.

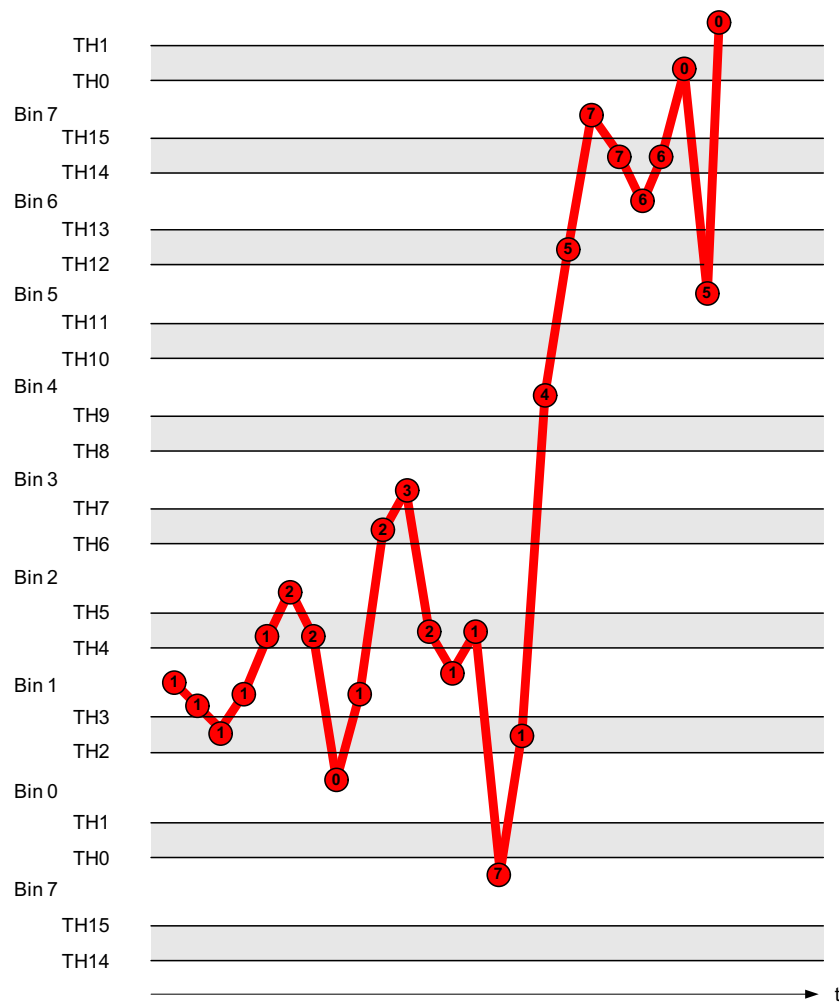


**Figure 30-7:** Hysteresis Example

If the pixel value falls in the gray region, the output index is computed based on the pseudo code below:

```
index = upper_bin
else if (previous_bin = lower_bin)
index = lower_bin
```



```
else if (previous_bin > upper_bin)
index = upper_bin
else
index = lower_bin
```

## Histogram Unit

The histogram unit has 16 histogram counters, each corresponding to one index. The counters run for the number of frames programmed in the PVP\_THCx\_HFCNT register. After the programmed number of frames the histogram counters are cleared and restarted. The histogram counters are moved to the PVP\_THCx\_HCNTy\_STAT registers at every frame. The value of the current frame count is also available in the PVP\_THCx\_HFCNT\_STAT register.

When the threshold unit is operating in hysteresis mode, the index output ranges from 0 to 7 and therefore only the PVP\_THCx\_HCNT0\_STAT to PVP\_THCx\_HCNT7\_STAT registers contain valid data. When the threshold unit is operating in clipping mode, the index output ranges from 0 to 2 and only the PVP\_THCx\_HCNT0\_STAT to PVP\_THCx\_HCNT2\_STAT registers have valid data. Each of the histogram counters is 32 bits wide. If the counter reaches the highest value of 0xFFFFFFFF before the expiration of the value in the PVP\_THCx\_HFCNT register, then the counter does not overflow but saturates at that value.

The histogram unit can be enabled or disabled using the PVP\_THCx\_CTL.HISTEN bit.

## Compression Unit

The compression unit uses the run-length-encoding (RLE) technique to compress data. In most cases the output from the threshold unit has a lot of repetition and converting the sequence of data values into a report containing value and run-length count reduces throughput at the output of the THC block.

The value and run-length are grouped into a single word based on the PVP\_THCx\_CTL.0FRMT bits. The run-length indicates the actual number of pixels and not repetitions, so the minimum run-length value is 1.

If the PVP\_THCx\_CTL.RLEFRAME bit is 0 then a report is generated at the end of a row and the run-length counter is reset. If = 1, then compression works across rows in the frame. Due to the nature of compression, a variable number of reports are generated per row or per frame. The PVP\_THCx\_RREP\_STAT register contains the actual number of reports generated per frame. A value can be set in the PVP\_THCx\_RMAXREP register to limit the number of reports generated per row or per frame. When the number of reports reaches the limit then the last report's run-length field is cleared.

Note that some formats in PVP\_THCx\_CTL.0FRMT bits disables compression. Also, some formats use angle information to generate the final report.

A typical usage of compression unit is by setting the maximum number of reports in PVP\_THCx\_RMAXREP register. Program the output DMA to transfer the maximum number of reports. Once all the reports are generated for the frame, the compression unit sends a *Finish Command* to DMA, ensuring that the work unit moves to the next work unit despite the variable data per work unit. The actual number of reports generated can be read from PVP\_THCx\_RREP\_STAT status register.

## Windowing

The histogram and compression units support windowing to filter incoming pixels.

The histogram window supports filtering pixels either inside or outside the window. The window mode is selected by setting the `PVP_THCx_CTL.HISTWM` bits. The window coordinates are specified by the `PVP_THCx_HHPOS`, `PVP_THCx_HVPOS`, `PVP_THCx_HHCNT` and `PVP_THCx_HVCNT` registers.

The compression window supports filtering pixels only inside the window. The window mode is selected by setting the `PVP_THCx_CTL.RLEWM` bits. The window coordinates are specified by the `PVP_THCx_RHPOS`, `PVP_THCx_RVPOS`, `PVP_THCx_RHCNT` and `PVP_THCx_RVCNT` registers. The minimum supported window size is 16x16.

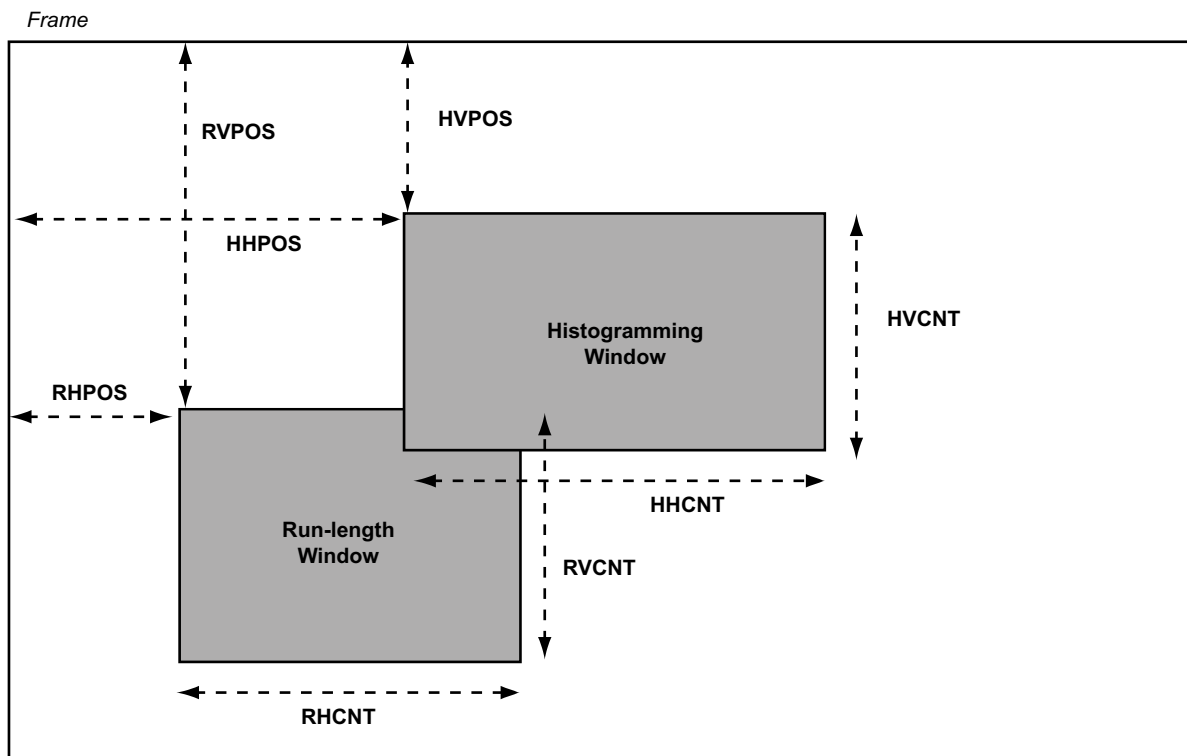


**Figure 30-8:** Definition of RLE and Histogram Windows

## Convolution (CNVx)

Convolution is a mathematical operation on two signals producing a third signal which is typically viewed as a modified version of one of the original signals. The PVP has four convolution blocks.

The convolution block implements a 2D convolution of the input pixels with a 5x5 coefficient matrix. The actual mathematical operation is described in the equation below.

$$Y_{i,j} \Big|_{i=0-(M-1), j=0-(N-1)} = \sum_{m=0-4, n=0-4} X_{i+m-2, j+n-2} * C_{m,n}$$

where:

- N = number of columns in the input data
- M = number of rows in the input data
- Y = MxN output data
- X = MxN input data
- C = 5x5 coefficient data

Note that the strict mathematical formula for convolution mirrors the coefficients in both directions before the MAC operation. For CNV blocks, the coefficients are multiplied in place to simplify the operation (similar to Correlation). Programs can always put the 5x5 coefficients in different order to realize either convolution or correlation.

The following figure shows the data flow within the CNV block.

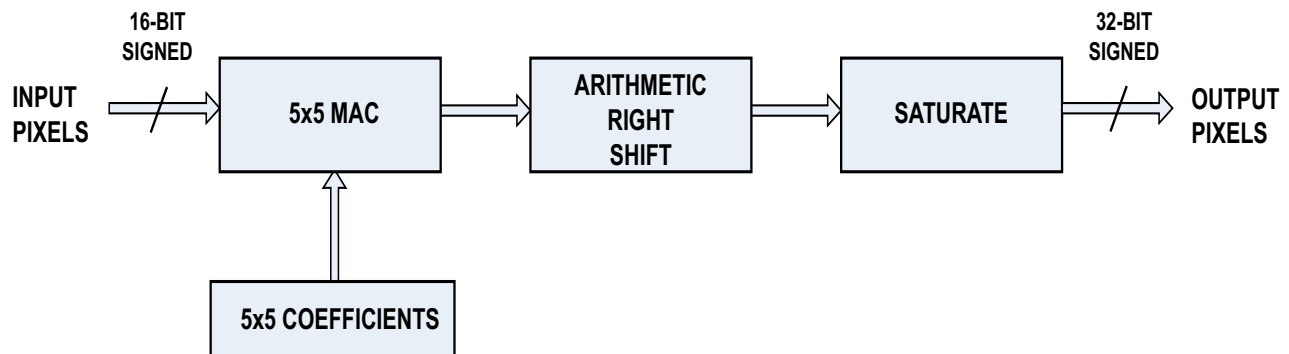


**Figure 30-9:** CNVx Overview

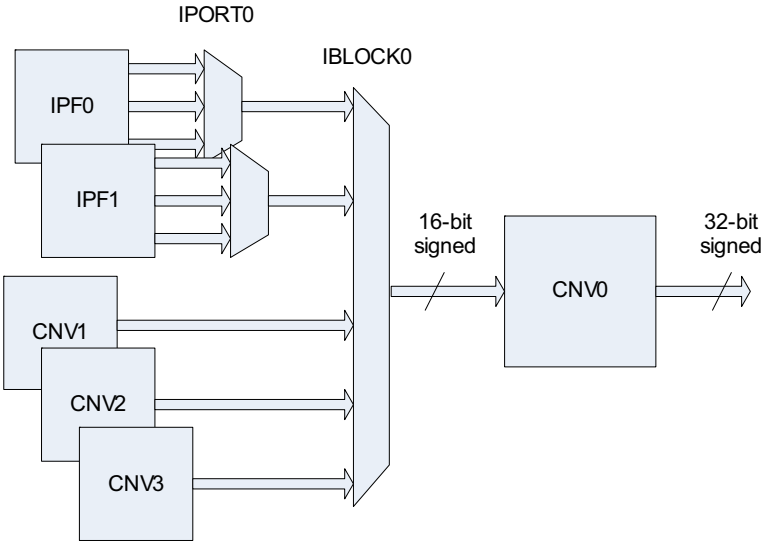


Figure 30-10: Convolution Block Connectivity

Table 30-13: CNV0 Block Connectivity

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| s16     |         |         |      | s32      |          |          |
| Block   | Port    | Block   | Port |          |          |          |
| IPF0    | 0, 1, 2 | n/a     | n/a  | Result   | n/a      | n/a      |
| IPF1    | 0, 1, 2 |         |      |          |          |          |
| CNV1    | 0       |         |      |          |          |          |
| CNV2    | 0       |         |      |          |          |          |
| CNV3    | 0       |         |      |          |          |          |

Table 30-14: CNV1 Block Connectivity

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| s16     |         |         |      | s32      |          |          |
| Block   | Port    | Block   | Port |          |          |          |
| IPF0    | 0, 1, 2 | n/a     | n/a  | Result   | n/a      | n/a      |
| IPF1    | 0, 1, 2 |         |      |          |          |          |
| CNV0    | 0       |         |      |          |          |          |
| CNV2    | 0       |         |      |          |          |          |
| CNV3    | 0       |         |      |          |          |          |

**Table 30-15:** CNV2 Block Connectivity

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| s16     |         |         |      | s32      |          |          |
| Block   | Port    | Block   | Port |          |          |          |
| IPF0    | 0, 1, 2 | n/a     | n/a  | Result   | n/a      | n/a      |
| IPF1    | 0, 1, 2 |         |      |          |          |          |
| CNV0    | 0       |         |      |          |          |          |
| CNV1    | 0       |         |      |          |          |          |
| CNV3    | 0       |         |      |          |          |          |

**Table 30-16:** CNV3 Block Connectivity

| Input 0 |         | Input 1 |      | Output 0 | Output 1 | Output 2 |
|---------|---------|---------|------|----------|----------|----------|
| s16     |         |         |      | s32      |          |          |
| Block   | Port    | Block   | Port |          |          |          |
| IPF0    | 0, 1, 2 | n/a     | n/a  | Result   | n/a      | n/a      |
| IPF1    | 0, 1, 2 |         |      |          |          |          |
| CNV0    | 0       |         |      |          |          |          |
| CNV1    | 0       |         |      |          |          |          |
| CNV2    | 0       |         |      |          |          |          |

The CNV block expects 16-bit signed input data and drives 32-bit signed output data. The key features of the block are:

- When the `PVP_CNVx_CTL.SAT32` bit = 0, the output is saturated to a 16-bit value and sign extended to drive the 32-bit output. When the bit = 1, the output is saturated to a 32-bit value.
- The boundary pixels at the output require input pixels beyond the edges for computation. When the `PVP_CNVx_CTL.ZEROFILL` bit = 1, the input pixels outside the edges are considered to be 0. When the bit = 0, the edges are duplicated beyond the boundary of the input pixels.
- The `PVP_CNVx_CTL.SHIFT` bit controls output arithmetic right shift from 0 to 31 bits. This allows the coefficients to be in fractional format with full control of the placement of the binary point.
- The CNV block supports downscaling by dropping output pixels in both directions. The `PVP_CNVx_SCALE.HSCL` bits are used to set the horizontal downscale factor and `PVP_CNVx_SCALE.VSCL` bits are used to set the vertical downscale factor. The block supports decimation in powers of 2. The maximum horizontal downscale factor is 1024 and the vertical factor is 512.

The value programmed in these registers is one less than the actual scale factor. Therefore a scale factor of 0 disables downscaling.

The `PVP_CNVx_SCALE.HSCL` and `PVP_CNVx_SCALE.VSCL` settings must not change on the fly if the output of the downscaling CNVx block supplies another CNVx block. If so, the `DRAIN` bit must be set for the new configuration.

- The CNV block supports 5x5 convolution kernels. The block automatically supports any smaller 2D kernels like 3x3 by centering the kernel in the 5x5 matrix and filling the surrounding values with zeros. The coefficients for the  $y^{\text{th}}$  row are set in the `PVP_CNVx_Cy0Cy1`, `PVP_CNVx_Cy2Cy3` and `PVP_CNVx_Cy4` registers.
- The accumulator in the CNV block is 37-bits wide—no overflow is possible during MAC operations.

**NOTE:** A downscaled CNV cannot drive another CNV. A CNV can only drive another CNV as long as it is not downscaled.

## Polar Magnitude and Angle Block (PMAx)

The PVP features one polar magnitude and angle block (PMA). The block takes two 16-bit signed inputs as coordinates in the Cartesian form (x, y) and converts them into Polar form (Magnitude, Angle  $\phi$ ).

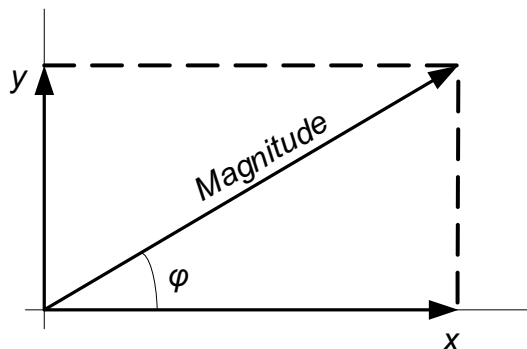


**Figure 30-11:** Cartesian vs. Polar form

The PMA block applies two different mathematical formulas (Pythagorean) to obtain magnitude and angle:

$$\text{Magnitude} = \sqrt{|x|^2 + |y|^2}$$

$$\phi = \arctan\left(\frac{y}{x}\right)$$

**Figure 30-12:** PMA Equations

The magnitude is output as a 16-bit unsigned value, ranging from 0 to the square root of 2. For  $x = y = 0x7FFF$ , it outputs `0xB503`. For the special case of  $x = y = 0x8000$ , the PMA outputs `0xB504`.

The angle  $\phi$  is calculated to five bits of resolution ( $11.25^\circ$ ) and an accuracy of  $\pm 0.25^\circ$ . Mathematically, the angle is a signed value. However, since the angle is always zero-extended when output to 16-bit or 32-bit buses, it can also be interpreted as an unsigned binning value.

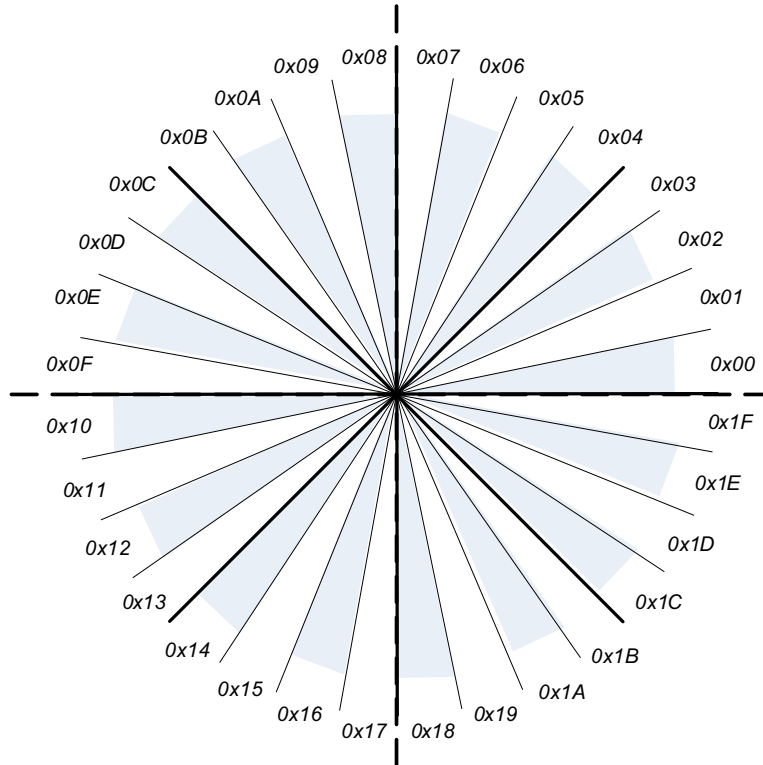


**Figure 30-13:** PMA Angle Binning

The PMA sets the angle to 0x08 ( $+90^\circ$ ) if the x input value is 0 and the y input is a positive value and to 0x18 ( $-90^\circ$ ) if the y input is negative. If both, x and y are zero, the PMA outputs 0x1F by convention.

Other than data flow configuration, the PMA block does not have any control or status registers. Both of its inputs can be individually configured to receive data from any of the convolution blocks or from either input formatter. Care is required in that both inputs are timed consistently and are not subject to non-matching latency in up front pipeline configuration.

**Table 30-17:** PMA Block Connectivity

| Input 0 | Block | Port    | Input 1 | Block | Port    | Output                                                             |
|---------|-------|---------|---------|-------|---------|--------------------------------------------------------------------|
| s16     | IPF0  | 0, 1, 2 | s16     | IPF0  | 0, 1, 2 | Port1: s32<br>magnitude<br>Port1: s32<br>angle Port2:<br>s32 combo |
|         | IPF1  | 0, 1, 2 |         | IPF1  | 0, 1, 2 |                                                                    |
|         | CNV0  | 0       |         | CNV0  | 0       |                                                                    |
|         | CNV1  | 0       |         | CNV1  | 0       |                                                                    |
|         | CNV2  | 0       |         | CNV2  | 0       |                                                                    |
|         | CNV3  | 0       |         | CNV3  | 0       |                                                                    |

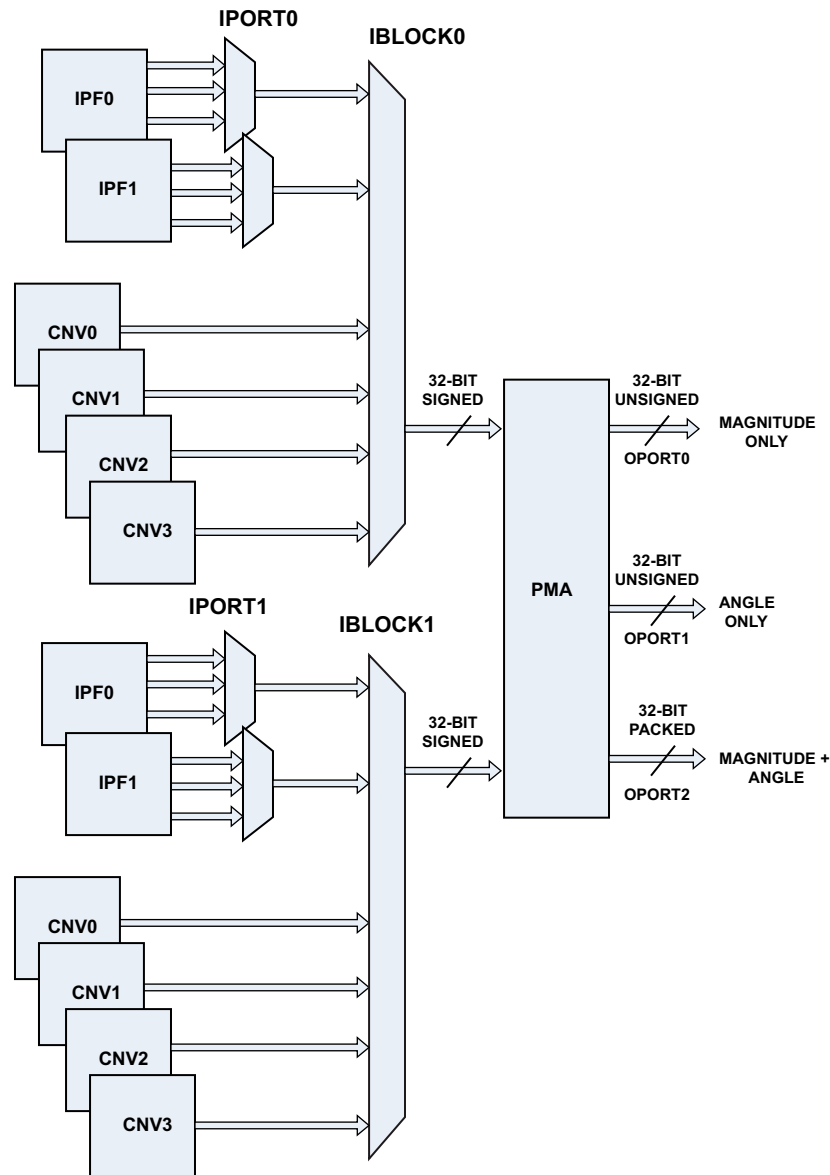


**Figure 30-14:** PMA Connectivity

The PMA block has three output ports. If PMA is enabled, all three 32-bit ports are always active as follows.

- Port 0 drives the 16-bit unsigned magnitude. The upper 16 bits are always driven as zeros.
- Port 1 drives the 5-bit angle value. The upper 27 bits are always driven as zeros.
- Port 2 drives a combined format. The lower 16 bits contain the magnitude, bits 16 to 20 drive the angle, and the upper ones are always zero. The Port 2 signal is not only good for being streamed to system memory. The THCx blocks have special functionality to deal with this format.



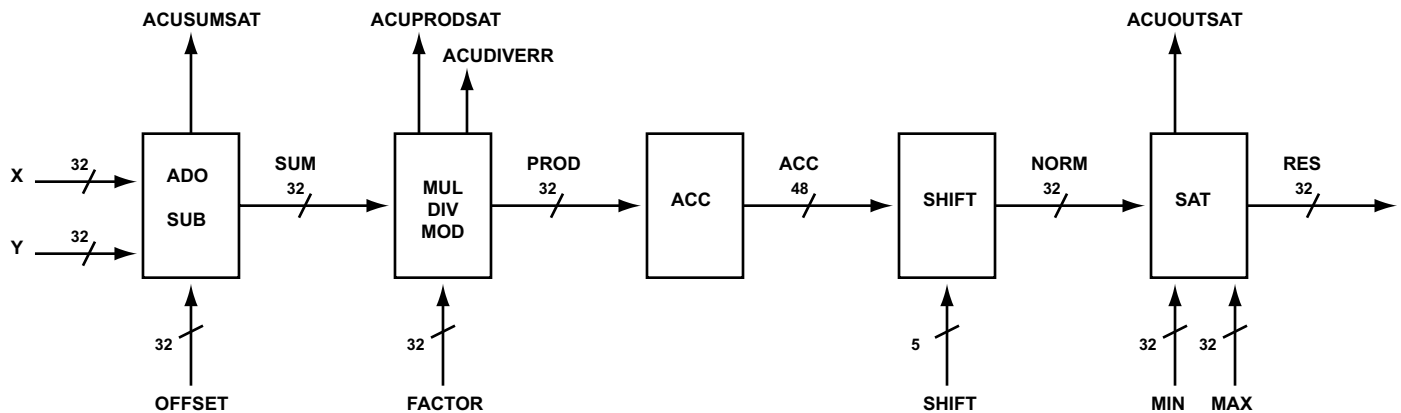
## Arithmetic Control Unit (ACUx)

The PVP features one arithmetic control unit (ACU). It provides basic 32-bit math like addition, subtraction, multiplication and division. It also features a 48-bit accumulator and can normalize and saturate on the output. While the ACU is important building block in many video applications, it also adds significant general-purpose value to the processor.

The ACU block has two 32-bit inputs (x, y) and one 32-bit output (res). It features a number of registers that hold constant operands. The *offset* value can be used for addition and subtraction. The *factor* value can be used for multiplication and division. The *shift* value is used for normalization and the *min* and *max* pair of registers control the saturation.

As shown in figure below, the individual operation can also be combined. Multiple computes can be performed in a single pass. For example,  $((x+y) \times 3) \gg 2$  can be performed by setting FACTOR = 3 and SHIFT = 2.

ACU Overview



For combinations of computes the individual operations need to be in the order shown above. For example  $((x \gg 2) \times 3) + y$  cannot be performed in a single pass. Similarly, not every operation can be done on every operand. The following table illustrates options available by individual math sub blocks.

**Table 30-18:** ACU Block Operations and Operands

| RESULT | OPERATIONS                                          | OPERANDS                                                         |
|--------|-----------------------------------------------------|------------------------------------------------------------------|
| SUM =  | + (32-bit signed add)<br>- (32-bit signed subtract) | X, Y<br>Y, X<br>X, OFFSET<br>OFFSET, X<br>Y, OFFSET<br>OFFSET, Y |

**Table 30-18:** ACU Block Operations and Operands (Continued)

| RESULT | OPERATIONS                                                                         | OPERANDS                                                                                       |
|--------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| PROD = | x (32-bit signed multiply)<br>/ (32-bit signed divide)<br>% (32-bit signed modulo) | X, Y<br>Y, X<br>X, FACTOR<br>FACTOR, X<br>Y, FACTOR<br>FACTOR, Y<br>SUM, FACTOR<br>FACTOR, SUM |
| ACC =  | 48-bit accumulate over row<br>48-bit accumulate over frame                         | X<br>SUM<br>PROD                                                                               |
| NORM = | 48-bit arithmetic right shift                                                      | X, SHIFT<br>SUM, SHIFT<br>PROD, SHIFT<br>ACC, SHIFT                                            |
| RES =  | 32-bit signed saturation                                                           | NORM, MIN, MAX                                                                                 |

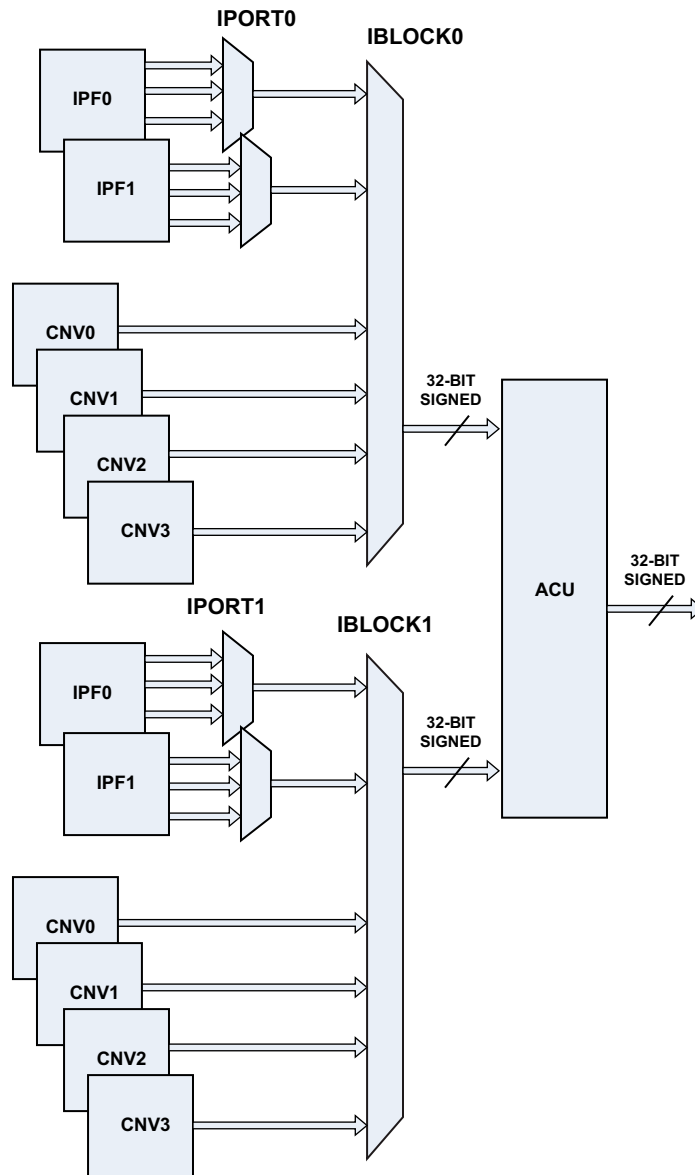


**Figure 30-15: ACU Connectivity**

For divide-by-zero operations, the quotient is set to the enumerator's value. The remainder is set to zero. The event is reported by the `PVP_STAT.ACUDIVERR` status bit. The operation itself progresses normally.

The intermediate results SUM and PROD are 32-bit values. If the result exceeds 32-bit range, the data is implicitly saturated to `0x7FFFFFFF` for positive results and to `0x80000000` for negative results. The `PVP_STAT.ACUSUMSAT` and `PVP_STAT.ACUPRODSAT` status bits report the saturation events.

If accumulation is active (`PVP_ACU0_CTL.SFTINP = 3`), the ACU outputs only one value every video row (`PVP_ACU0_CTL.ACCFRAME = 0`) or every frame (`PVP_ACU0_CTL.ACCFRAME = 1`). With every new row or frame, the accumulator is cleared automatically. The 32-bit input values or the intermediate 32-bit SUM or PROD results are summed up into the 48-bit accumulator. If intermediate accumulation results exceed

48 bits before the end of row/frame the accumulator silently overflows. It neither saturates nor generates a status event.

The SHIFT and the SAT operations are not optional. Their functionality can be bypassed by leaving the default zero value in the PVP\_ACU0\_SHIFT register (no shift) and by setting the clipping values in the PVP\_ACU0\_MIN register to 0x80000000 and in the PVP\_ACU0\_MAX register to 0x7FFFFFFF. Saturation events due to MAX/MIN register method are reported by the PVP\_STAT.ACU0OUTSAT status bit.

The ACU can receive data from either input formatter or any convolution block. Both inputs can be individually configured. Care is required to properly time input data so that both inputs match.

**Table 30-19:** ACU Block Connectivity

| Input 0 | Block | Port    | Input 1 | Block | Port    | Output |
|---------|-------|---------|---------|-------|---------|--------|
| s32     | IPF0  | 0, 1, 2 | s32     | IPF0  | 0, 1, 2 | s32    |
|         | IPF1  | 0, 1, 2 |         | IPF1  | 0, 1, 2 |        |
|         | CNV0  | 0       |         | CNV0  | 0       |        |
|         | CNV1  | 0       |         | CNV1  | 0       |        |
|         | CNV2  | 0       |         | CNV2  | 0       |        |
|         | CNV3  | 0       |         | CNV3  | 0       |        |

If the X input doesn't influence the ACU output based on the PVP\_ACU\_CTL settings, then the PVP\_ACU\_CFG.IBLOCK0 bit must be set to zero. Similarly, if Y is not used in the equation, the PVP\_ACU\_CFG.IBLOCK1 bit must be zero.

Input values can be doubled using the adder, if X and Y inputs sense that same data. Similarly, input values can be squared. The intermediate SUM result, however, cannot be squared.

The ACU is very useful in memory pipe mode when the supported operation needs to apply to all elements in an array. If one operand is a constant value as in  $res_i = x_i \% c$ , then the constant  $c$  value is best written into the constant register, (the PVP\_ACU\_FACTOR register in the example). Input 0 ( $x$ ) is typically connected to port 0 of IPF1. If, however, the operation is applied to elements of two arrays as in  $res_i = x_i / y_i$ , then, input 0 ( $x$ ) typically connects to port 1 of IPF1 and input 1 ( $y$ ) connects to port 2. The IPF expects the input data in interleaved manner, ( $x_0, y_0, x_1, y_1, x_2, y_2, \dots$ ). If data is not already stored this way in memory, the 2-dimensional feature of the DMA can perform the interleaving.

## Pixel Edge Classifier (PEC)

The PEC block processes the edge information to enhance the features for further vision processing. The PEC supports two modes of operation—1st derivative mode or PEC-1 and 2nd derivative mode or PEC-2. These modes are mutually exclusive and the PEC can operate in only one mode at a time. The PEC block takes 32-bit input data and drive up to three streams of output data.

PEC-1 supports the subsequent stages after the Sobel filters for the first derivative Canny edge detection algorithm. The primary features of PEC-1 are:

- Edge thinning feature using non-maximum suppression.
- Streaking elimination using hysteresis (3-level thresholding on the magnitude of the gradient).

In PEC-1 mode, the block expects the magnitude on the lower 16-bits and the angle information on the upper 16-bits. Therefore, the PEC block can either interface to the PMA output port-2 which has the packed magnitude and angle, or the IPF ports, assuming the IPF drives the packed magnitude, angle data.

The PEC-2 can generate 8-connected chain codes for contour tracing using output from second derivative edge detection methods like Difference of Gaussian (DoG) and Laplacian of Gaussian (LoG). The primary features of PEC-2 are:

- Detection of zero-crossing points in the second derivative filter output.
- Classification of the edges based on the edge direction.
- Compute location of the edge with half pixel resolution (sub-pixel interpolation).
- The PEC block supports back-to-back frame size change in output from CNV block. The block supports incoming frames with a decimation ratio of up to 256 (1, 2, 4, 8, 16, 32, 64, 128 and 256). The smallest frame size that the PEC can support is 16x16.
- Both PEC-1 and PEC-2 output zeros for pixels on the frame boundaries.
- The PEC only supports even sized frames.
- The PEC mode cannot change from frame to frame. The block has to be disabled and enabled to change mode.

In PEC-2 mode, the block expects 16-bit signed input data. Therefore, the PEC block can interface with either the IPF ports, the ACU block (32-bit signed, but saturated to 16-bits) or any of the CNV blocks (32-bit signed, but saturated to 16-bits). In this mode the PEC only operates on the lower 16 bits of the 32-bit signed input. Figure x shows the high level connectivity of the PEC block.

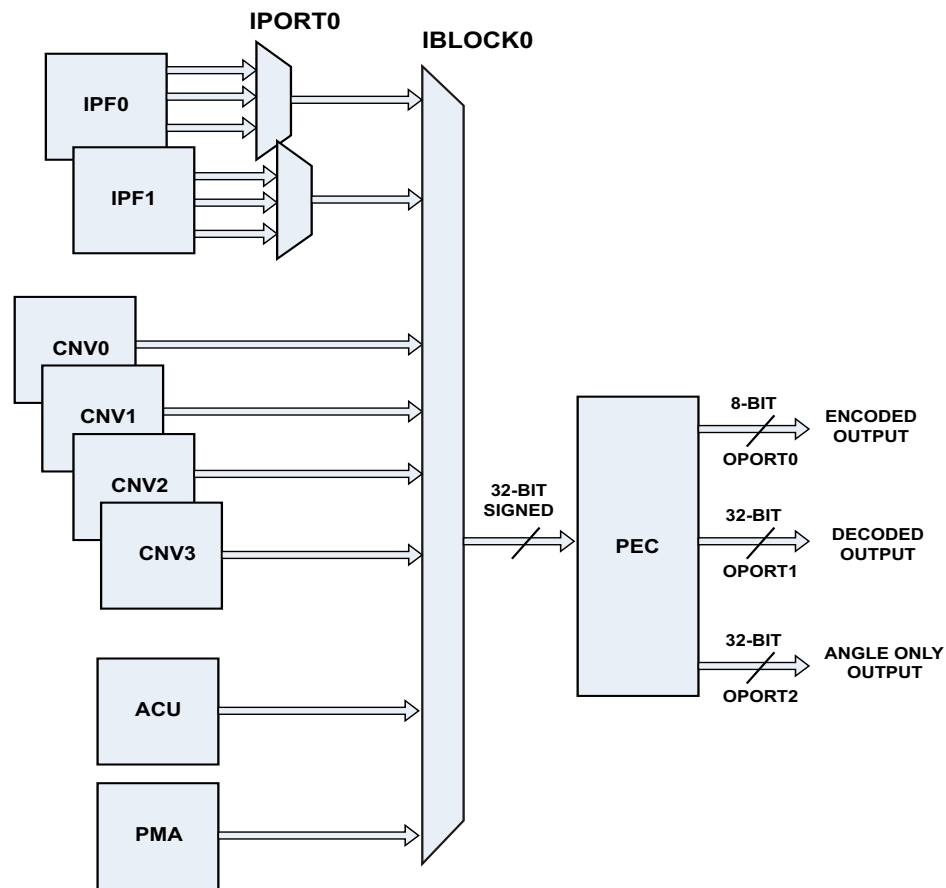


Figure 30-16: Pixel Edge Classifier Block Connectivity

The details of the connectivity of the PEC block are described in the table below.

Table 30-20: PEC Block Connectivity

| Input 0 |         | Input 1 |      | Output 0       | Output 1       | Output 2          |
|---------|---------|---------|------|----------------|----------------|-------------------|
| s32     |         |         |      | u8             | u32            | u32               |
| Block   | Port    | Block   | Port |                |                |                   |
| IPF0    | 0, 1, 2 | n/a     | n/a  | encoded result | decoded result | angle only result |
| IPF1    | 0, 1, 2 |         |      |                |                |                   |
| PMA     | 2       |         |      |                |                |                   |
| CNV0    | 0       |         |      |                |                |                   |
| CNV1    | 0       |         |      |                |                |                   |
| CNV2    | 0       |         |      |                |                |                   |
| CNV3    | 0       |         |      |                |                |                   |
| ACU     | 0       |         |      |                |                |                   |

## PEC 1st Derivative Mode (PEC-1)

This mode is selected by setting the `PVP_PEC_CTL.MODE` bit to 0 and works on 16-bit gradient magnitude and 5-bit gradient angle from the PMA block.

### Angle Mapping

The angle is rounded to one of four angles representing vertical, horizontal and the two diagonals. The 5-bit angle bin output of the PMA is mapped to the four directions as shown in the following figure.

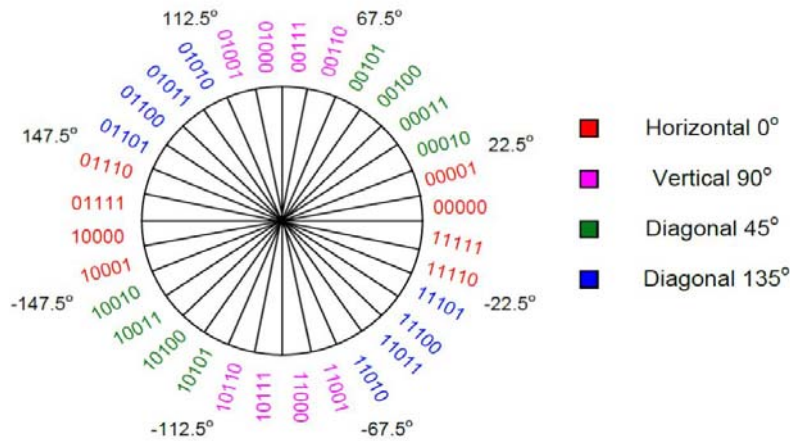


**Figure 30-17:** PMA Angle Bin Mapping to PEC-1 Angle

### Non-Maximum Suppression

A search is performed with the input magnitude gradient to determine if the magnitude assumes a local maximum in the gradient direction so that thin edges are produced. This is referred to as *Non-Maximum Suppression*.

The PEC-1 works on the 3x3 window with the center pixel as the pixel of interest and computes whether the pixel is a local maximum or not. Consider a 3x3 window of magnitude gradients as shown below.

|     |     |     |
|-----|-----|-----|
| P11 | P12 | P13 |
| P21 | P22 | P23 |
| P31 | P32 | P33 |

The pixel of interest (P22) is *maxima* (M),

- If the rounded angle is 0 degrees and if its intensity is greater than the intensity in the west (P21) and greater than the intensity in the east, (P23) directions.
- If the rounded angle is 90 degrees and if its intensity is greater than the intensity in the north (P12) and greater than the intensity in the south (P32) directions.
- If the rounded angle is 135 degrees and if its intensity is greater than the intensity in the north west (P11) and greater than the intensity in the south east (P33) directions.
- If the rounded angle is 45 degrees and if its intensity is greater than the intensity in the north east (P13) and greater than the intensity in the south west (P31) directions.

In cases where the gradient magnitude is the same for more than one pixel along the gradient direction “equality” (E) information is also provided by PEC-1. Edge-tracing software uses M and E to find the exact location of the edge. Consider the same 3x3 window as before.

The pixel of interest (P22) is *equality* (E),

- If the rounded angle is zero degrees and if its intensity is greater than or equal to the intensity in the west (P21) and greater than or equal to the intensity in the east, (P23) directions.
- If the rounded angle is 90 degrees and if its intensity is greater than or equal to the intensity in the north (P12) and greater than or equal to the intensity in the south (P32) directions.
- If the rounded angle is 135 degrees and if its intensity is greater than or equal to the intensity in the north west (P11) and greater than or equal to the intensity in the south east (P33) directions.
- If the rounded angle is 45 degrees and if its intensity is greater than or equal to the intensity in the north east (P13) and greater than or equal to the intensity in the south west (P31) directions.

## Hysteresis Thresholding

The Canny algorithm uses *hysteresis* thresholding. When a single threshold value is used, the fluctuations of the gradient magnitude (due to noise) above and below the threshold value results in pixels being classified as edges and non-edges. In such cases the edge line appears broken. This phenomenon is commonly referred to as streaking. PEC-1 eliminates streaking by comparing the magnitude against two threshold registers, PVP\_PEC\_D1TH0 and PVP\_PEC\_D1TH1.

If the magnitude lies below PVP\_PEC\_D1TH0 (TL), then the pixel is called *no edge*. If the magnitude lies above PVP\_PEC\_D1TH1 (TH) then the pixel is termed as *strong edge*. If it lies between TL and TH, then the pixel is termed as *weak edge*. The post-processing software can connect the weak edge to strong edge to create a unbroken edge line. Note that TH must be programmed higher than TL for hysteresis.

## Output Format

The PEC-1 produces three different outputs on three ports. The details are described below.



## Port0

This output is primarily used for post-processing in software. Encoded 8-bit output is produced for every input pixel as shown below.

| bit 7     | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 | bit 0 |
|-----------|-------|-------|-------|-------|-------|-------|-------|
| Angle Bin |       |       |       |       | Code  |       |       |

The 5-bit angle bin from the PMA/IPF is passed on to the 5 MSBs of the PEC-1 output if the pixel magnitude is greater than or equal to TL. If the pixel magnitude is less than TL, the 5 MSBs are suppressed to zeros. PEC-1 combines the maxima (M), equality (E) and hysteresis thresholding to produce a 3-bit code output (which forms the 3 LSBs of the PEC-1 output) whose bits are defined as shown below.]

**Table 30-21:** PEC-1 Code Mapping

| LSB of PEC Output | Description                   |
|-------------------|-------------------------------|
| 000               | mag < TL                      |
| 001               | mag >= TL, mag < TH, M=0, E=0 |
| 010               | mag >= TL, mag < TH, M=0, E=1 |
| 011               | mag >= TL, mag < TH, M=1, E=1 |
| 100               | reserved                      |
| 101               | mag >= TH, M=0, E=0           |
| 110               | mag >= TH, M=0, E=1           |
| 111               | mag >= TH, M=1, E=1           |

## Port1

This output is 32 bits wide and is used primarily for the IIM block to accumulate the edge statistics. The output can be either 8-bit bin size encoded or 16-bit bin size encoded based on the setting of the PVP\_PEC\_CTL.OSIZE control bit. If bit is 0 (indicating 8-bit bin size), the 32-bit output represents four angle bins, each having a strong/weak edge information as shown in the table below.

**Table 30-22:** Output Encoding for 8-bit Bin Size

| Angle    | Edge    | Encoding                            |
|----------|---------|-------------------------------------|
| Mag < TL | No edge | 00000000 00000000 00000000 00000000 |

**Table 30-22:** Output Encoding for 8-bit Bin Size (Continued)

| Angle        | Edge   | Encoding                            |
|--------------|--------|-------------------------------------|
| 0° or 180°   | Weak   | 00000000 00000000 00000000 00000001 |
| 0° or 180°   | Strong | 00000000 00000000 00000000 00000010 |
| 45° or 225°  | Weak   | 00000000 00000000 00000001 00000000 |
| 45° or 225°  | Strong | 00000000 00000000 00000010 00000000 |
| 90° or 270°  | Weak   | 00000000 00000001 00000000 00000000 |
| 90° or 270°  | Strong | 00000000 00000010 00000000 00000000 |
| 135° or 315° | Weak   | 00000001 00000000 00000000 00000000 |
| 135° or 315° | Strong | 00000010 00000000 00000000 00000000 |

If the PVP\_PEC\_CTL.0SIZE bit is 1 (indicating 16-bit bin size), the 32-bit output represents two angle bins, each having a strong/weak edge information as shown in the table below.

**Table 30-23:** Output Encoding for 16-bit Bin Size

| Angle                       | Edge    | Encoding                            |
|-----------------------------|---------|-------------------------------------|
| Mag < TL                    | No edge | 00000000 00000000 00000000 00000000 |
| 0° or 180° or 90° or 270°   | Weak    | 00000000 00000000 00000000 00000001 |
| 0° or 180° or 90° or 270°   | Strong  | 00000000 00000000 00000000 00000010 |
| 45° or 225° or 135° or 315° | Weak    | 00000000 00000001 00000000 00000000 |
| 45° or 225° or 135° or 315° | Strong  | 00000000 00000010 00000000 00000000 |

The strong/weak edge is determined by the threshold values programmed in the PVP\_PEC\_D1TH0 (TL) and PVP\_PEC\_D1TH1 (TH) registers. The PVP\_PEC\_CTL.IGNTH1 bit supports masking the differences between strong and weak edges. If this bit is 0, weak and strong edges are treated differently and the output is as per the tables above. If this bit is 1, all the edges are considered weak irrespective of the TH value and the output is encoded to represent a weak edge as per the above tables.

## Port2

This output is 5 bits wide and is used primarily for generating the angle histograms. The 5-bit angle bin from the PMA/IPF is passed on to this port if the pixel magnitude is greater than or equal to the

PVP\_PEC\_D1TH0 register. If the center pixel magnitude is less than PVP\_PEC\_D1TH0, this output is suppressed to zeros.

## PEC 2nd Derivative Mode (PEC-2)

This mode is selected by setting the PVP\_PEC\_CTL.MODE bit to 1. PEC-2 works on 16-bit signed data from the second derivative filter output like DoG and LoG. The primary functions in this mode follow.

### Zero-Crossing Detection

Edges lie on the zero crossing points (-ve to +ve and +ve to -ve transition points) in the 2nd derivative filter output. The PEC-2 scans through the filtered image and determines zero crossings in horizontal (zH) and vertical (zV) directions and assigns pre-defined 3-bit zero crossing codes. The zero crossing detection algorithm is explained below.

Whenever there is a +ve to -ve or -ve to +ve in the filtered image, the pixel with +ve value in the image is considered to be a valid zero-crossing point. As the 2nd derivative filtered image is more prone to errors due to noise, a user defined threshold is used to remove the noise during zero crossing detection. Zero crossings are considered valid only if there is a sign change and the difference between the 2nd derivative values of these two pixels is greater than the threshold in the PVP\_PEC\_D2TH0 register. When the pixel of interest is has a value of zero, then it is considered as a valid zero-crossing point if the two pixels on the left and right (for horizontal and top, bottom for vertical) are values with opposite signs, (+ 0 -) or (- 0 +). The MSB of the 3-bit code is set if the difference between the pixels is greater than the threshold in the PVP\_PEC\_D2\_TH1 register.

The following table shows the different combinations of 2nd derivative image values for the pixel of interest, the pixel on the left and the pixel on the right and the corresponding 3-bit zero-crossing codes for horizontal direction (zH). The vertical zero-crossings (zV) are obtained by replacing left pixel with the top pixel and the right pixel with the bottom pixel in the same table.

**Table 30-24:** Zero Crossing Codes

| Left       | Center | Right      | Computation                                                                                                                   | zH (3-bit)        |
|------------|--------|------------|-------------------------------------------------------------------------------------------------------------------------------|-------------------|
| Don't care | -ve    | Don't care |                                                                                                                               | 000               |
| +ve / 0    | +ve    | +ve / 0    |                                                                                                                               | 000               |
| -ve / 0    | 0      | 0 / -ve    |                                                                                                                               | 000               |
| -ve        | +ve    | +ve / 0    | if (center-left >= Threshold_weak) &<br>(center-left < Threshold_strong)<br>else if (center-left >= Threshold_strong)<br>else | 010<br>110<br>000 |

**Table 30-24:** Zero Crossing Codes (Continued)

| Left    | Center | Right   | Computation                                                                                                                                                                                                                                                                                    | zH (3-bit)                                      |
|---------|--------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
| -ve     | +ve    | -ve     | if (center-left $\geq$ Threshold_weak) &<br>(center-left < Threshold_strong)<br><br>else if (center-left $\geq$ Threshold_strong)<br><br>else if (center-right $\geq$ Threshold_weak) &<br>(center-right < Threshold_strong)<br><br>else if (center-right $\geq$ Threshold_strong)<br><br>else | 010<br><br>110<br><br>001<br><br>101<br><br>000 |
| +ve / 0 | +ve    | -ve     | if (center-right $\geq$ Threshold_weak) &<br>(center-right < Threshold_strong)<br><br>else if (center-right $\geq$ Threshold_strong)<br>else                                                                                                                                                   | 001<br><br>101<br><br>000                       |
| +ve / 0 | 0      | +ve / 0 |                                                                                                                                                                                                                                                                                                | 000                                             |
| -ve     | 0      | +ve     | if (right-left $\geq$ 2 $\times$ Threshold_weak) &<br>(right-left < 2 $\times$ Threshold_strong)<br><br>else if (right-left $\geq$ 2 $\times$ Threshold_strong)<br><br>else                                                                                                                    | 010<br><br>110<br><br>000                       |
| +ve     | 0      | -ve     | if (left-right $\geq$ 2 $\times$ Threshold_weak) &<br>(left-right < 2 $\times$ Threshold_strong)<br><br>else if (left-right $\geq$ 2 $\times$ Threshold_strong)<br><br>else                                                                                                                    | 001<br><br>101<br><br>000                       |
| -ve     | 0      | -ve     |                                                                                                                                                                                                                                                                                                | 000                                             |

## Sub-Pixel Interpolation

A linear interpolation of the filtered image at the zero crossings is performed to determine a more precise location of the zero crossings. The values of (left, center, right) for horizontal direction and (top, center,

bottom) for vertical direction are compared for relative differences. Additionally, edge location with half pixel resolution is computed.

Table 19 explains the different zero crossing codes, pixel value sequences and the corresponding sub-pixel values (HSub, 2-bits). The vertical sub-pixel values (VSub, 2-bits) can be computed using the same procedure, replacing left pixel with top pixel, right pixel with bottom pixel and the 2 LSBs of zH with zV.

**Table 30-25:** Sub-Pixel Computation

| zH (2-bits) | Left       | Center     | Right      | Computation                   | Sub-pixel<br>HSub (2-bits) |
|-------------|------------|------------|------------|-------------------------------|----------------------------|
| 00          | Don't care | Don't care | Don't care | -                             | 00                         |
| 10/01       | Don't care | 0          | Don't care | -                             | 00                         |
| 10          | -          | +          | Don't care | if (-left) <= center<br>else  | 10 00                      |
| 01          | Don't care | +          | -          | if (-right) <= center<br>else | 01<br>00                   |

## Angle Classification

The 2nd derivative filtering of an image doesn't convey any information about the orientation of the edges in the image. The angle classification unit in PEC-2 extracts the angle information of the edges from the zero crossing codes of pixels in the 3x3 neighborhood. The classifier uses only the two lower bits of zH and zV, (both weak and strong zero crossings). The classifier classifies the edges into eight basic directions (0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°) if the PVP\_PEC\_CTL.ZCRSS bit is 0. The classifier outputs a 4-bit angle index as described in the table below.

**Table 30-26:** Angle Classification

| Priority | Zero Crossing Code Combination                                                         | Class          | Angle Index (4-bit) |
|----------|----------------------------------------------------------------------------------------|----------------|---------------------|
| 1        | $zH(r,c) = 00 \ \& \ zV(r,c) = 00$                                                     | no edge        | 0000                |
| 2        | Not matching any of the combination 4 to 11.                                           | no match       | 0001                |
| 3        | Matching more than one combination in 4 to 11.                                         | multiple match | 0010                |
| 4        | $zV(r,c) = 01 \ \& \ zV(r,c+1) = 01$                                                   | 0°             | 1000                |
| 5        | $(zH(r,c) = 01 \    \ zV(r,c) = 01) \ \& \ (zH(r-1,c+1) = 01 \    \ zV(r-1,c+1) = 01)$ | 45°            | 1001                |
| 6        | $zH(r,c) = 01 \ \& \ zH(r-1,c) = 01$                                                   | 90°            | 1010                |

**Table 30-26:** Angle Classification (Continued)

| Priority | Zero Crossing Code Combination                                                                | Class | Angle Index (4-bit) |
|----------|-----------------------------------------------------------------------------------------------|-------|---------------------|
| 7        | $u$<br>$(zH(r-1,c-1) = 01 \parallel zV(r-1,c-1) = 10)$                                        | 135°  | 1011                |
| 8        | $zV(r,c) = 10 \& zV(r,c-1) = 10$                                                              | 180°  | 1100                |
| 9        | $(zH(r,c) = 10 \parallel zV(r,c) = 10) \&$<br>$(zH(r+1,c-1) = 10 \parallel zV(r+1,c-1) = 10)$ | 225°  | 1101                |
| 10       | $zH(r,c) = 10 \& zH(r+1,c) = 10$                                                              | 270°  | 1110                |
| 11       | $(zH(r,c) = 10 \parallel zV(r,c) = 01) \&$<br>$(zH(r+1,c+1) = 10 \parallel zV(r+1,c+1) = 01)$ | 315°  | 1111                |

The angle index generated above is considered a strong edge only if all the involved zH or zV values used in the computation are strong (or have their MSB = 1). If the PVP\_PEC\_CTL.IGNTH1 bit is set then the strong/weak zero-crossing information (MSB of zH and zV) is ignored by the classifier.

## Output Format

The PEC-2 produces three different outputs on three ports as described below.

Port0

This output is used primarily for post-processing in software. The 8-bit encoded output is produced for every input pixel as shown below.

| PVP_PEC_CTL.ZCRS | bit 7 | bit 6 | bit 5 | bit 4 | bit 3       | bit 2 | bit 1 | bit 0 |
|------------------|-------|-------|-------|-------|-------------|-------|-------|-------|
| 0                | VSub  |       | HSub  |       | Angle Index |       |       |       |
| 1                | VSub  |       | HSub  |       | zV          |       | zH    |       |

Port1

This output is 32-bits wide and is used primarily for the IIM block to accumulate the edge statistics. The output is not valid when the PVP\_PEC\_CTL.ZCRSS bit = 1 since the edges are not classified in this mode. PEC-2 produces a 32-bit output which is exactly similar to the output generated by PEC-1. The angle is computed as shown in the [Angle Classification](#) table. For priority 1, 2, and 3, the output is considered as no edge and all zeros are driven on the port.

Port2

This output is 5-bits wide and is used primarily for the THCx block to generate the angle histograms. PEC-2 drives the 4-bit angle index described in the [Angle Classification](#) table. Zero is driven on the MSB of the 5-bit port. For priority 1, 2, and 3, the output is considered as no edge and all zeros are driven on the port. The output is not valid when the PVP\_PEC\_CTL.ZCRSS bit = 1 since the edges are not classified in this mode.

## Integral Image Block (IIMx)

The PVP features two integral image blocks (IIM). The IIMs calculate a 2-dimensional integral over the input frame and outputs the summed area table (SAT). Alternatively, the IIM blocks can generate a rotated SAT (RSAT) or can integrate in horizontal dimension only (integral row mode). The operating mode is controlled by the PVP\_IIMx\_CTL.MODE bit field.

### Integral Row Mode

In integral row mode the IIM blocks accumulate the input values over video rows. Unlike the ACU's accumulation mode, integral row mode outputs the integral every pixel. For every input data  $Inp_{I,J}$  the IIM blocks output is as shown in the following equation.

$$IROW_{I,J} = \sum_{i=0}^{I-1} Inp_{i,J} \quad IROW_{I,J} = Inp_{I,J} + IROW_{I-1,J}$$

The initial value of the integral is assumed to be zero. Therefore,  $IROW_{0,J} = Inp_{0,J}$ . In the IROW output map, each value contains the accumulated input values from the respective row's origin to the actual position as shown in the following example.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 0 | 2 | 3 | 0 |
| 0 | 0 | 1 | 0 | 3 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 2 | 0 | 1 | 0 | 5 |
| 1 | 0 | 0 | 1 | 0 | 0 | 2 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

Input Data:  
 $0+2+3+1+0+1+0 = 7$

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 3 | 4 | 4 | 6 | 9 | 9 |
| 0 | 0 | 1 | 1 | 4 | 4 | 4 | 4 |
| 0 | 2 | 5 | 6 | 6 | 7 | 7 | 7 |
| 0 | 0 | 1 | 3 | 3 | 4 | 4 | 9 |
| 1 | 1 | 1 | 2 | 2 | 2 | 4 | 4 |
| 1 | 1 | 2 | 2 | 2 | 3 | 4 | 4 |

Integral Row Mode

**Figure 30-18:** Illustration of Integral Row Mode

### Summed Area Table (SAT) Mode

In 2-dimensional SAT mode, the following equation is performed for every pixel.

$$SAT_{I,J} = \sum_{j=0}^{J-1} \sum_{i=0}^{I-1} Inp_{i,j} \quad SAT_{I,J} = Inp_{I,J} + SAT_{I-1,J} + SAT_{I,J-1} - SAT_{I-1,J-1}$$

Again, initial values of the integral are assumed to be zero— $SAT_{-1,J} = 0$  and  $SAT_{I,-1} = 0$ . In the SAT output map, each value contains the accumulated input values from the frame's origin to the actual position as shown in the figure below.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 2 | 0 | 1 | 1 | 0 | 2 | 3 | 0 |
| 0 | 0 | 1 | 0 | 3 | 0 | 0 | 0 |
| 0 | 2 | 3 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 2 | 0 | 1 | 0 | 5 |
| 1 | 0 | 0 | 1 | 0 | 0 | 2 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |

*Input Data:*  
 $2+0+1+1+0+0+1+0+0+2+3+1 = 11$

|   |   |    |    |    |    |    |    |
|---|---|----|----|----|----|----|----|
| 2 | 2 | 3  | 4  | 4  | 6  | 9  | 9  |
| 2 | 2 | 4  | 5  | 8  | 10 | 13 | 13 |
| 2 | 4 | 9  | 11 | 14 | 17 | 20 | 20 |
| 2 | 4 | 10 | 14 | 17 | 21 | 24 | 29 |
| 3 | 5 | 11 | 16 | 19 | 23 | 28 | 33 |
| 4 | 6 | 13 | 18 | 21 | 26 | 32 | 37 |

*SAT Mode*

**Figure 30-19:** Illustration of SAT Mode

## Rotated Summed Area Table (RSAT) Mode

In RSAT mode a  $-45^\circ$  rotated view is applied to the input coordinates. Each element of the RSAT output map contains the integral of all elements in the pyramid headed by the actual position. The IIM engine process according to the following procedure.

First Column:

$$RSAT_{0,J} = Inp_{0,J} + Inp_{0,J-1} + RSAT_{1,J-1}$$

Last Column:

$$RSAT_{I,J} = Inp_{I,J} + Inp_{I,J-1} + RSAT_{I-1,J-1}$$

All others:

$$RSAT_{I,J} = Inp_{I,J} + Inp_{I,J-1} + RSAT_{I+1,J-1} + RSAT_{I-1,J-1} - RSAT_{I,J-2}$$

All values outside of the region are defined as zero— $Inp_{I,-1} = RSAT_{I,-1} = RSAT_{I,-2} = 0$ .



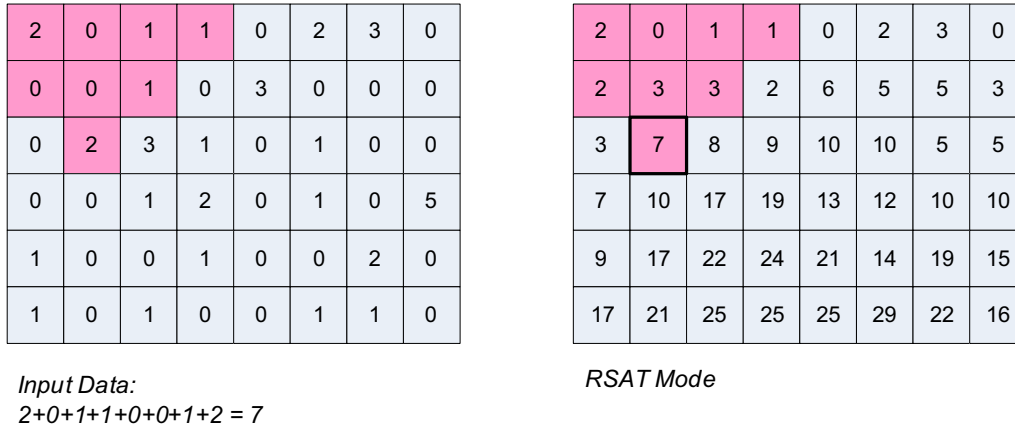


**Figure 30-20:** Illustration of RSAT Mode

## Using SAT and RSAT Maps

The utility of the integral is not only that each position contains the sum of all input values between the actual position and the frame origin. It is that with only a small number of operations, post-processing software can determine the integral value of any rectangular region of the frame. For regular rectangles the SAT output can be used. For 45° rotated rectangles the RSAT output applies. This is illustrated in the following figure.

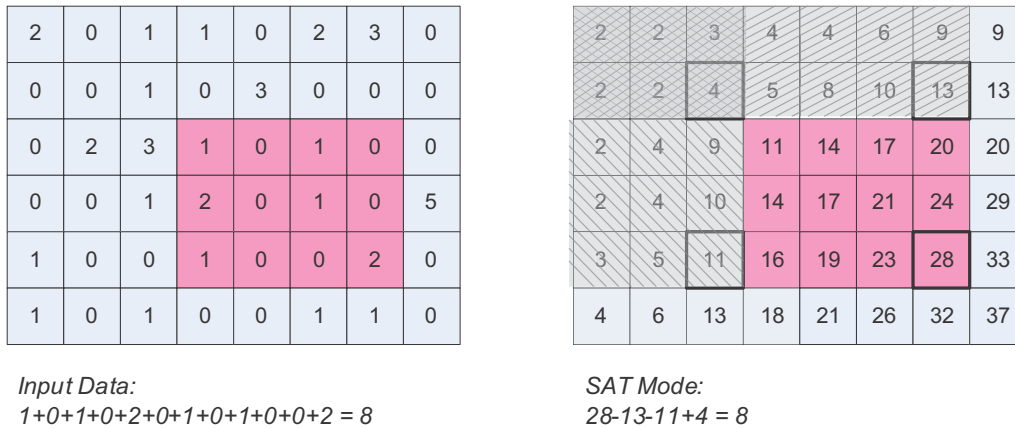


**Figure 30-21:** Using SAT Maps

Rather than summing up all original values of a rectangular region of the frame, software can perform simple math on the SAT output of the IIM blocks. If the top-left corner of the rectangle was defined by the coordinates  $I_0$  and  $J_0$ , and the bottom-right corner by  $I_1$  and  $J_1$ , then the integral value of the region is defined by the following formula.

$$\iint_{\text{Region}} \text{Inp}_{i,j} = \sum_{j=J_0}^{J_1} \sum_{i=I_0}^{I_1} \text{Inp}_{i,j} = \text{SAT}_{I_1,J_1} - \text{SAT}_{I_1,J_0-1} - \text{SAT}_{I_0-1,J_1} + \text{SAT}_{I_0-1,J_0-1}$$

Similarly, the RSAT map can be used to quickly determine the integral of rotated regions as shown in the following figure.

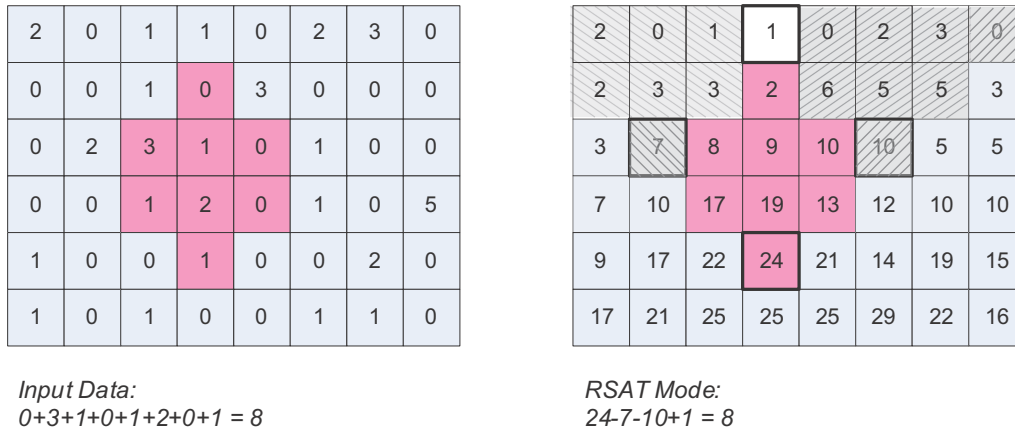


**Figure 30-22:** Use of RSAT Map

## Data Types

IIM blocks feature a 33-bit adder to operate on 32-bit data. The 33rd bit is used for overflow management. If the adder overflows, it does not saturate but continues normal operation. Software may post-analyze the integral map to track overflows. The occurrence of an overflow depends on the input data format. For 10-bit input data, the integral counter should never overflow over the region or even the full frame.

The occurrence of overflows can be controlled by reducing the resolution of the incoming data. The `PVP_IIMx_CTL.SHIFT` control field instructs the IIMx block to arithmetically right shift input data. It can shift from 0 to 31 bit positions to the right.

The IIM blocks can operate on signed and unsigned data. There is no control bit to distinguish between signed and unsigned mode. Rather, there are two different sets of status reports. The event of unsigned overflow is reported by the `PVP_STAT.IIMxUOVF` bits. When flagged, the `PVP_IIMx_UOVF_STAT` register latches the coordinates of the data that caused the overflow. Similarly, the `PVP_STAT.IIMxSOVF` bits report signed overflows and the `PVP_IIMx_SOVF_STAT` register latches the coordinates.

The adder can be configured to operate in dual 16-bit mode or in quad 8-bit mode, alternatively. This is controlled by the `PVP_IIMx_CTL.WIDTH` field. In these modes each IIMx block can perform dual or quad integral operations concurrently, for the price of having increased likelihood of overflows. The overflow reporting mechanism works the same as in the 32-bit mode where it monitors overflows in the topmost word or byte. In dual 16-bit mode or in quad 8-bit mode the input shifter performs an unsigned right shift operation rather than an arithmetical operation as in 32-bit mode.

The quad 8-bit mode is most useful when the integral block connects to port 1 of the PEC. In this case the IIMx generates the integral over the 1st or 2nd derivative gradients. Dual 16-bit mode might connect to port 2 of the PMA. In this case the lower part of the adder builds the integral over the magnitude while the upper half integrates the angle. Alternatively, port 2 of the PMA might be routed through one of the THCx blocks for quantization of the magnitude.

## Controlling Bandwidth

In many cases the IIMx blocks receive input data with a relatively low bit count. The output however, is always 32 bits wide. If not used with care the IIMx blocks can be significant consumers of system memory bandwidth. If the 32-bit resolution is not required at the output, the connected output formatter may only take the upper or the lower 16 bits of the result. To take the lower bits, set `PVP_OPFx_CTL.ISIZE = 1` and `PVP_OPFx_CTL.IUP16 = 0`. To take the upper bits, set `PVP_OPFx_CTL.ISIZE = 1` and `PVP_OPFx_CTL.IUP16 = 1`. Together with the `PVP_IIMx_CTL.SHIFT` setting the weight of the integral can be controlled.

Another way of reducing bandwidth used by the integral blocks is by scaling the IIMx output. The `PVP_IIMx_SCALE` register can individually enable sub-sampling of rows or columns on the output in power-of-2 steps as follows.

- If `VSCL = 0`, no vertical scaling is performed and all rows are output.
- If `VSCL = 1`, the IIMx outputs only every other row and if `VSCL = 3`, it outputs only every 4th row.
- If `HSCL = 7`, the IIMx outputs only every 8th column.

## Connectivity

Both integral blocks can receive their inputs from a large variety of modules. They can build the integral on raw data inputs when receiving data from the input formatters. Input data can optionally be filtered by convolutions. Variances can be calculated by squaring the inputs by the ACU before building the integral. If connected to the PMA blocks, integral of edge magnitude and gradients can be built. Histogram of gradients can be done by sensing port 1 of the PEC block. If connected to the THC<sub>x</sub> blocks, the integral of a binary edge map can be generated, or data is clipped up front by the THC<sub>x</sub> blocks.

**Table 30-27:** IIMx Block Connectivity

| Input 0                                                     | Block                                                                             | Port                                                               | Input 1 | Block | Port | Output                                                      |
|-------------------------------------------------------------|-----------------------------------------------------------------------------------|--------------------------------------------------------------------|---------|-------|------|-------------------------------------------------------------|
| 1x s32,<br>1x u32,<br>2x s16,<br>2x u16,<br>4x s8,<br>4x u8 | IPF0<br>IPF1<br>CNV0<br>CNV1<br>CNV2<br>CNV3<br>ACU<br>PEC<br>PMA<br>THC0<br>THC1 | 0, 1, 2<br>0, 1, 2<br>0<br>0<br>0<br>0<br>0<br>1<br>0, 1<br>0<br>0 | n/a     | 0     | 0    | 1x s32,<br>1x u32,<br>2x s16,<br>2x u16,<br>4x s8,<br>4x u8 |

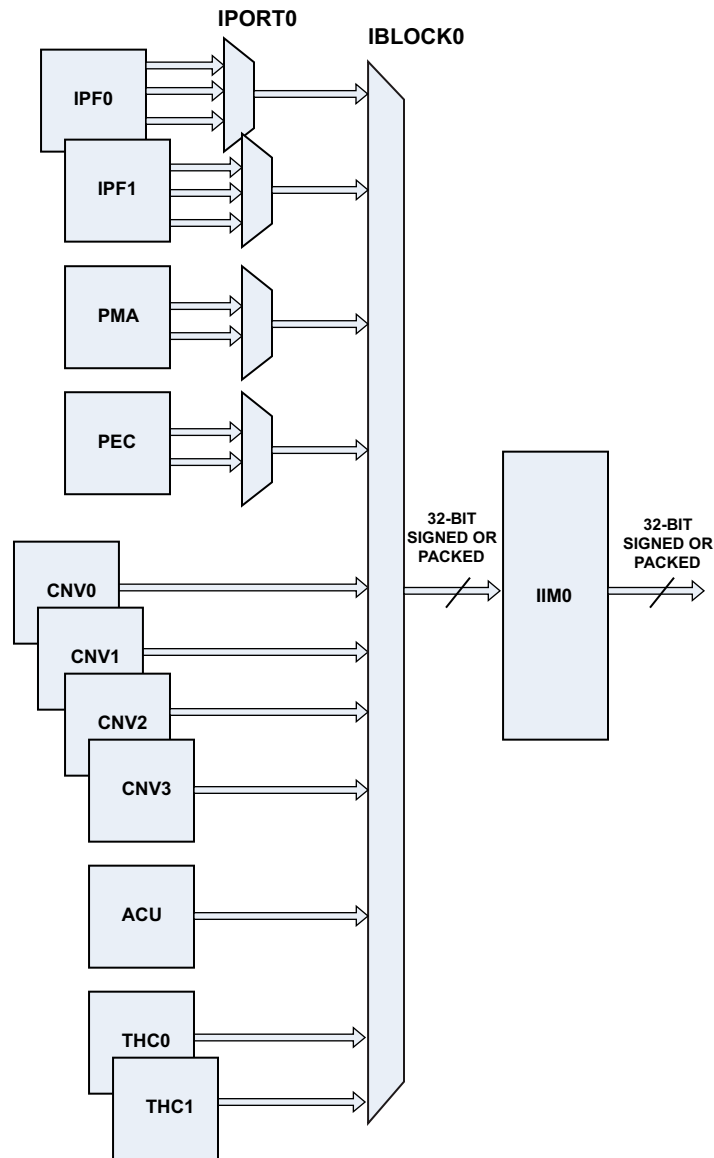


**Figure 30-23:** IIM0 Connectivity

## Up Down Scaler (UDS)

The UDS is an image resizing block. The UDS is the only block which cannot operate in the camera pipe. The input to UDS comes from IPF1 while the output drives OPF3 as shown in the figure below.

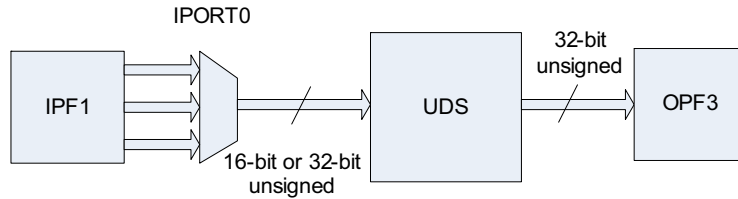


**Figure 30-24:** UDS Block Connections

The UDS block expects 16-bit or 32-bit unsigned input data and drives 32-bit unsigned output data. When an anti-aliasing or an averaging filter is enabled, the input must be 16 bits and correspondingly the output is 16 bits presented in the lower 16 bits of the 32-bit output.

The figure below shows the data flow within the UDS block. The block performs horizontal scaling followed by vertical scaling in two independent steps. The interpolation uses bilinear arithmetic.

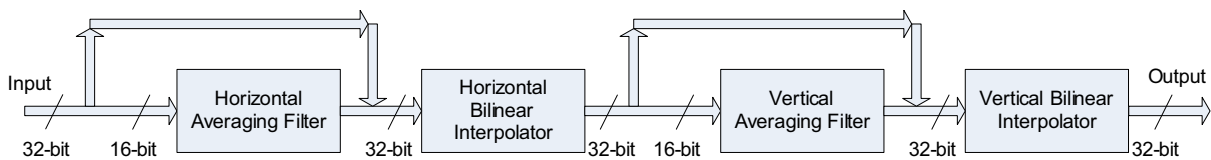


**Figure 30-25:** UDS Overview

The key features of the block are as follows.

- Supports input frame sizes from 10x10 to 1280x960. The frame size is programmed in the PVP\_IPF1\_HCNT and PVP\_IPF1\_VCNT registers of IPF1.
- Supports independent scaling in both directions. The output frame size is selectable in steps of 16. The values supported are:  $16 \times n \times 16 \times m$  where  $n, m = 1, 2, \dots, 8$ . The output size is programmed in the PVP\_UDS\_OHCNT and PVP\_UDS\_OVCNT registers.
- Supports an optional anti-aliasing filter in the form of an averaging filter in both directions. The filter has an option to calculate the number of taps automatically or takes a user programmable value. If the PVP\_UDS\_CTL.AAVG bit = 1 then the number of taps are selected automatically based on the input and output frame size. If the bit = 0 then the number of taps must be programmed in the PVP\_UDS\_HAVG and PVP\_UDS\_VAVG registers. The UDS supports a maximum of 128 taps for horizontal direction and 64 taps for vertical direction.
- Supports an averaging filter only for 16-bit input data. The averaging filter must be disabled for 32-bit input data by setting the PVP\_UDS\_CTL.AAVG bit = 0, the PVP\_UDS\_HAVG.VALUE = 1, and the PVP\_UDS\_VAVG.VALUE bit = 1. The filter is bypassed when the number of taps is 1.

The number of taps for each direction in automatic mode is calculated as:

$$N\_TAPS = \left\lfloor \frac{Source\_Dimension}{Target\_Dimension} \right\rfloor$$

where  $\lfloor x \rfloor$  denotes integer part of x. In user programmable mode the number of taps is limited by the following equation.

$$N\_TAPS \leq \left( 2 * \frac{Source\_Dimension}{Target\_Dimension} \right) - 1$$

If an output pixel requires input pixels beyond the edges then the nearest pixel on the edge is used for computation. In effect edge pixels are duplicated from the nearest valid pixel.

## Architectural Concepts

The PVP unifies two architectural concepts, the camera pipes and the memory pipe. The camera pipes serve the preprocessing requirements. Data is preprocessed on its way from the PPI input to the system memory and can be broadcast from one single video source to up to three functional branches called camera pipes. The memory pipe functions as a Co-processor. It loads input data from system memory via DMA and stores results back to system memory the same way.

Camera pipes and memory pipes can operate concurrently. They have completely separate control mechanisms and data paths and are completely independent in timing. Both pipes share, however, twelve signal processing blocks. The user has full flexibility in assigning each of the processing blocks to any pipe. The one exception is the up/down-scaler block that can only operate in memory pipe mode. Certainly, each block can be assigned only once at a given time. However, there is support for dynamic re configuration of the assignments. The blocks are:

- Two input formatters (IPFx, x=0, 1) supporting selection of input streams, gating pipeline processing clock, extraction of color or luminance components, windowing region of interest, separation of odd and even bit stream vectors (pixels) and frame counting. IPF0 receives input data from the video subsystem and feeds data to the camera pipes. IPF1 receives data from memory via DMA and services the Memory Pipe.
- Four convolution blocks (CNVx, x=0, 1, 2, 3) supporting 2-dimensional 5x5 convolution of data streams and constant coefficients, vector downscaling, data shifting, and data normalization.
- One polar magnitude and angle block (PMA) supporting data conversion from Cartesian form (X, Y) to a polar form (magnitude, angle).
- One pixel edge classifier (PEC) supporting edge detection in 2-dimensional data arrays using 2-dimensional 1<sup>st</sup> or 2<sup>nd</sup> derivation.
- One arithmetic unit (ACU) supporting 32-bit arithmetic operations (product, sum, accumulation, shift, maximum, minimum).

- Two threshold-histogram-compression cells (THCx, x=0, 1) supporting data formatting (clipping, saturation), data quantization (binning), multi-level hysteresis and data compression.
- Two integral image blocks (IIMx, x=0, 1) supporting integrals of rows and diagonals.
- One Up-down scaler (UDS) supporting interpolation and extrapolation of 2-dimensional data arrays.
- Four output formatters (OPFx, x=0, 1, 2, 3) supporting selection of output streams, data packing, and buffering (FIFO) output data streams. They collect the processing results and forward them to dedicated DMA channels.

## Operating Modes

Following sections provide operating mode examples, describing how the PVP can be used to implement typical vision-system operations.

### Thresholds and Histograms

The THCx block can be used for a variety of purposes. One example where the input data range is reduced to 4-bits and computation of 16-level histogram is shown in the figure below.

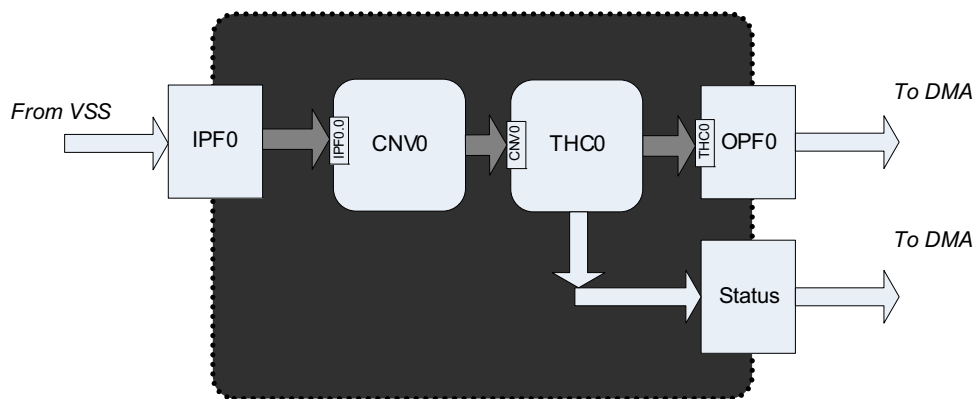


**Figure 30-26:** Computation of 16-level Histogram

In camera pipe mode, CNV0 receives data from port 0 of input formatter IPF0. The CNV0 block can do low pass filtering to eliminate noise and can optionally down scale the input for the THC0 block.

The THC0 block operates in the quantization mode by setting the `PVP_THC0_CTL.MODE` bit = 1. The threshold values are set in the `PVP_THC0_Thy` (y = 0–15) registers. The block is programmed to output a 4-bit index without any compression when the `PVP_THC0_CTL.OFRMT` bit = 2.

The histogram function is enabled by setting the `PVP_THC0_CTL.HISTEN` bit = 1. The number of frames used to perform the histogram operation is set in the `PVP_THC0_CTL.HFCNT.VALUE` bit field.

The OPF0 is set to pack the 4-bit output from THC0 to a 32-bit word when the `PVP_OPF0_CTL.ISIZE` bit = 3 and the `PVP_OPF0_CTL.OSIZE` bit = 0. The histogram values are read using the status DMA channel.

The values can also be read by directly by accessing the PVP\_THCO\_HCNT<sub>y</sub>\_STAT (y = 0–15) registers. The PVP\_THCO\_HFCNT\_STAT register indicates the current frame counter number to which the histogram values have been accumulated.

## Sobel with 3x3 or 5x5 Matrix Operation

The following figure shows a simple Sobel use case example. The Sobel algorithm is a standard algorithm used for edge detection applications. One convolution block is used to determine the horizontal component of 1st-derivative edge gradients, and another convolution block determines the vertical component. In camera pipe mode, both convolution blocks receive data from port 0 of input formatter IPF0.

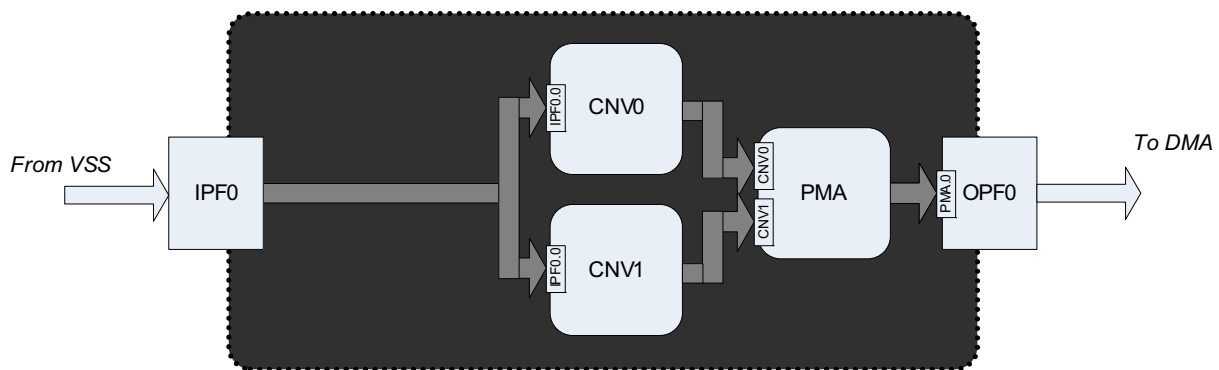


**Figure 30-27:** Sobel Data Flow

Traditional Sobel operators require simple 3x3 convolution matrices. The 5x5 coefficient matrices of the convolution blocks might be filled with 16-bit values as shown below.

$$Coeff_{CNV0} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 2 & 0 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad Coeff_{CNV1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

**Figure 30-28:** Traditional 3x3 Sobel Coefficients

Larger Sobel kernels can be realized using 5x5 operators. One option is the following set of coefficients.



$$Coeff_{CNV0} = \begin{pmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -8 & -4 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -8 & -4 \\ 1 & 2 & 0 & -2 & -1 \end{pmatrix} \quad Coeff_{CNV1} = \begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & -12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

**Figure 30-29:** Example of 5x5 Coefficients

The inputs of the subsequent PMA blocks should be 16-bit signed data. Therefore, care is required in dealing with data formats and bit growth. If IPF0 receives 14-bit unsigned data that is zero-extended to 16-bits, the 3x3 Sobel operation has a bit growth of 2 in each convolution block's accumulator. Additionally, the result is signed, making it a 17-bit signed value. Therefore, both convolution outputs need to be normalized by shifting two positions to the right (`PVP_CNVx_CTL.SHIFT = 2`). Alternatively, the accumulation results can be saturated to 16-bit signed values if no shift is performed and the `PVP_CNVx_CTL.SAT32` bit was kept clear.

If the input formatter receives 16-bit signed values and the above 5x5 coefficient matrix is used, then the accumulator holds 22-bit signed values and a `PVP_CNVx_CTL.SHIFT = 6` setting is required.

$$Magnitude = \sqrt{|x|^2 + |y|^2} \quad \varphi = \arctan\left(\frac{y}{x}\right)$$

**Figure 30-30:** PMA Operation

The PMA block takes the 16-bit signed Cartesian values from the two convolution operations. The PMA block converts the inputs to Polar values resulting in magnitude and the angle of the resulting gradients. The magnitude is a 16-bit unsigned format, and the angle is a 5-bit unsigned value.

## Sobel Output Formats

The PMA block has three output ports. Port 0 outputs the 16-bit unsigned magnitude, port 1 outputs the 5-bit angle and port 2 outputs the combination of the two: the lower 16 bits contain the magnitude and bits 20:16 contain the angle. All three ports zero fill the upper bits if connected to 32-bit data sinks.

If a system is only interested in the 16-bit magnitude values, then one of the output formatters is programmed to input data from PMA port 0 or from port 2. Likewise, the output formatter uses 16-to-32-bit packing and sets `PVP_OPFx_CTL.ISIZE = 1`, `PVP_OPFx_CTL.IUP16 = 0` and `PVP_OPFx_CTL.OSIZE = 0`.

If a system is only interested in the 5-bit magnitude values, then one of the output formatters is programmed to input data from PMA port 1. Likewise, the output formatter uses 8-to-32-bit packing and

sets `PVP_OPFx_CTL.ISIZE = 2`, `PVP_OPFx_CTL.IUP16 = 0` and `PVP_OPFx_CTL.OSIZE = 0`. Alternatively, the OPF can take data from port 2 and set `PVP_OPFx_CTL.IUP16 = 1`.

If, however, the system wants to use both magnitude and angle values, two output formatters can be used. However, for best system resource use it is recommended to only use one output formatter that inputs data from PMA port 2 in 32-bit mode where (`PVP_OPFx_CTL.ISIZE = 0`).

Often, an application does not require 16 bits of magnitude resolution. Rather, only one bit (binary edge map) or only a few bits are wanted. In this case, the threshold blocks are used as shown in the following figure.

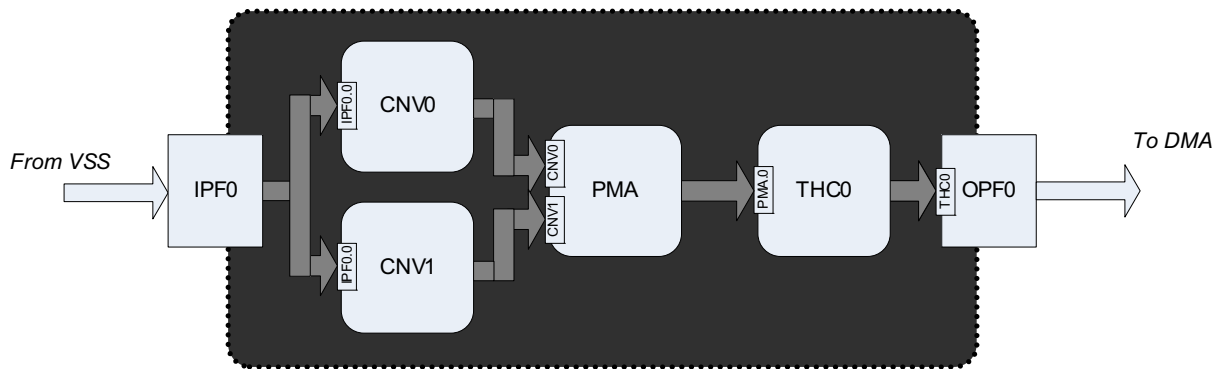


**Figure 30-31:** Thresholded Edge Map Data Flow

A binary edge map can be generated by the THC<sub>x</sub> using quantization mode. The `PVP_THCx_TH0` register is set to zero and the `PVP_THCx_TH1` register defines the threshold. If the output format is set to the 4-bit floor value (`PVP_THCx_CTL.OFRMT = 1`), a 4-bit magnitude value is sent. The connected output formatter may perform 4-to-32-bit packing by the setting the `PVP_OPFx_CTL.ISIZE` bit = 3. All 16 thresholds can be configured to apply a non-linear quantization of the magnitude value. Alternatively, hysteresis mode can be used. Clipping mode, when paired with `PVP_THCx_CTL.OFRMT = 0`, can be used to output user-programmable 32-bit values when the magnitude is above the threshold.

On the input side, the THC<sub>x</sub> blocks provide multiple options: if they sense Port 0 of the PMA block, they receive 16-bit values that are zero-extended to 32 bits. If they sense Port 2 instead, the 5-bits of might be unwanted. Then, the `PVP_THCx_CTL.ZEXT = 1` setting helps masking the upper 16 bits. This feature becomes most interesting when partnering with the `PVP_THCx_CTL.OFRMT = 4` option. This way, the THC generates an output format that groups the 4-bit floor value of the magnitude with the four significant bits of the PMA angle. The Output Formatter is best set to `PVP_OPFx_CTL.ISIZE = 2` mode.

## Canny with PEC in 1st-Derivative Mode

The following figure shows a Canny use case example. The Canny algorithm is a standard algorithm for edge detection applications. The input pixels are first low pass filtered to remove noise by CNV0. One convolution block is used to determine the horizontal component of 1st-derivative edge gradients, and another convolution blocks determines the vertical component as in the Sobel case. In camera pipe mode, the CNV0 block receives data from port 0 of IPF0.

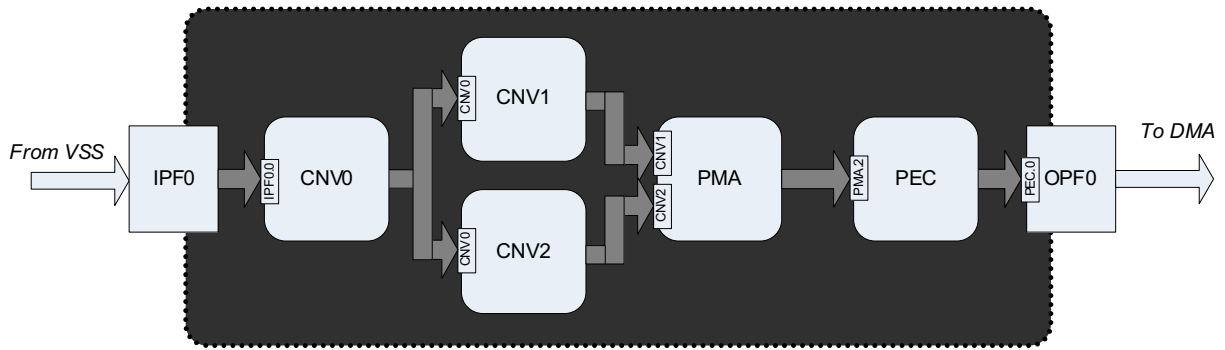


**Figure 30-32:** Canny Data Flow

The 5x5 Gaussian kernel (Sigma = 1) for low pass filtering is shown in the figure below.

$$Gaussian\ Kernel = \frac{1}{330} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 20 & 33 & 20 & 4 \\ 7 & 33 & 54 & 33 & 7 \\ 4 & 20 & 33 & 20 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \quad Coeff_{CNV0} = \begin{bmatrix} 0x0063 & 0x018d & 0x02b7 & 0x018d & 0x0063 \\ 0x018d & 0x07c2 & 0x0ccd & 0x07c2 & 0x018d \\ 0x02b7 & 0x0ccd & 0x14f2 & 0x0ccd & 0x02b7 \\ 0x018d & 0x07c2 & 0x0ccd & 0x07c2 & 0x018d \\ 0x0063 & 0x018d & 0x02b7 & 0x018d & 0x0063 \end{bmatrix}$$

**Figure 30-33:** Gauss Filter

The fractional values are converted to 1.15 format and programmed into the coefficient registers of CNV0. The output is shifted right by 15 bits to remove the fractional part by setting `PVP_CNV0_CTL.SHIFT` to 15.

The low pass filtered output feeds the two convolution blocks which generate the gradients as described in the Sobel use case example. The PMA block takes the 16-bit signed Cartesian values from the two convolution operations. The PMA block converts the inputs to polar values that result in magnitude and the angle of the resulting gradients. The magnitude is in a 16-bit unsigned format, and the angle is a 5-bit unsigned value.

The PEC block interfaces to PMA port 2 which has the magnitude and angle packed into a 32-bit word. The PEC performs the non-maximum suppression to produce thin edges and hysteresis threshold to allow software to reduce streaking. Since the PEC works on the output from 1st derivative filters, it should be configured for mode 1 by setting `PVP_PEC_CTL.MODE` to 0. The threshold values for magnitude comparison is set in `PVP_PEC_D1TH0` register (lower threshold) and the `PVP_PEC_D1TH1` register (higher threshold). Hysteresis thresholding can be selectively disabled by setting `PVP_PEC_CTL.IGNTH1` to 1.

The OPF can be set to sense PEC port 0 which carries the 8-bit encoded output. The post-processing software can use this output for edge tracing and further processing. The OPF should be configured to `PVP_OPF0_CTL.ISIZE = 2` and `PVP_OPF0_CTL.OSIZE = 0`.

## LoG with PEC in 2nd-Derivative Mode

Laplacian on Gaussian (LoG) is an edge detection algorithm based on 2nd derivative filtering of an input image. The LoG method produces thin edges but is sensitive to noise. Therefore the input image is filtered

using a Gaussian kernel before convolving with the LoG kernel. The PEC block produces the angle information which is missing in the traditional LoG algorithm. The post-processing software can use the class/angle information from the PEC to build contours for further analysis.

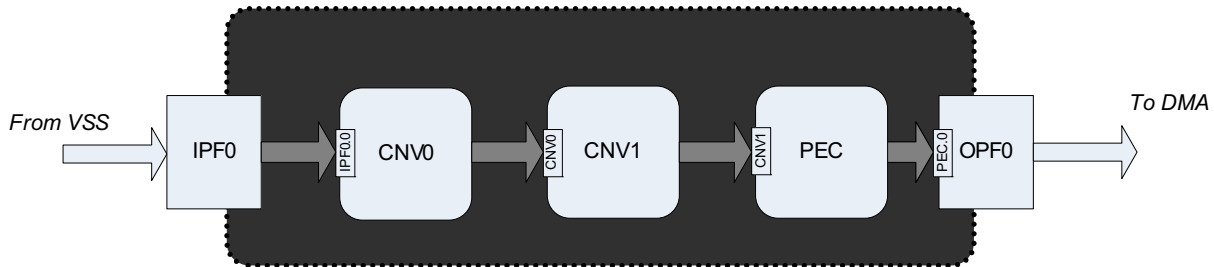


**Figure 30-34:** LoG Data Flow

The CNV0 use the same Gaussian kernel described in the Canny use case example. The LoG kernel is shown in the figure below.

$$LoG\ Kernel = \frac{1}{16} \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \quad Coeff_{CNV1} = \begin{bmatrix} 0 & 0 & 0xf800 & 0 & 0 \\ 0 & 0xf800 & 0xf000 & 0xf800 & 0 \\ 0xf800 & 0xf000 & 0x7fff & 0xf000 & 0xf800 \\ 0 & 0xf800 & 0xf000 & 0xf800 & 0 \\ 0 & 0 & 0xf800 & 0 & 0 \end{bmatrix}$$

**Figure 30-35:** LoG Coefficients

The coefficients are programmed in 1.15 format in the CNV1 registers. The fractional part is eliminated in the output of CNV1 by setting `PVP_CNV1_CTL.SHIFT = 15`.

The PEC block interfaces to the CNV1 output. The PEC computes angle information and sub-pixel location of the edges. Since the PEC works on outputs from 2nd derivative filters, it should be configured to mode 2 by setting `PVP_PEC_CTL.MODE = 1`. The threshold values for zero crossing detection is set in the `PVP_PEC_D2TH0` (lower threshold) and `PVP_PEC_D2TH1` (higher threshold) registers. The strong/weak edge distinction can be selectively disabled by setting `PVP_PEC_CTL.IGNTH1 = 1`. The PEC can output either the zero-crossing codes (`PVP_PEC_CTL.ZCRSS = 1`) or the angle information (`PVP_PEC_CTL.ZCRSS = 0`) from the classifier. In both cases the output on port 0 is in 8-bit packed format and contains the sub-pixel location in the upper 4-bits and either zero-cross code or angle index.

The OPF can be set to sense port 0 of the PEC which carries the 8-bit encoded output. The post-processing software can use this output for building contours and further processing. The OPF should be configured to `PVP_OPF0_CTL.ISIZE = 2` and `PVP_OPF0_CTL.OSIZE = 0`.

The following figure demonstrates how both 1st derivative and 2nd derivative edge detection methods can be realized concurrently in the PVP.

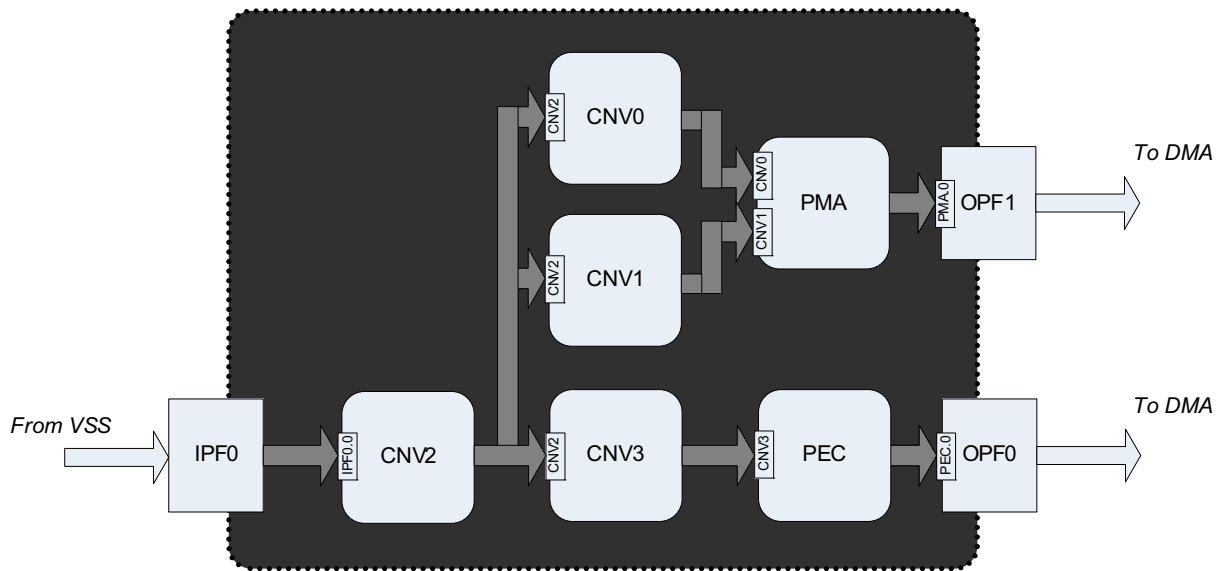


**Figure 30-36:** Concurrency of Sobel and LoG Methods

The CNV2 performs the initial low-pass filtering to remove noise in the input data. The output is broadcast to the three CNV blocks. The lower signal pipe using CNV3-PEC realizes the 2nd derivative method of edge detection using the LoG method. The upper signal pipe realized the 1st derivative method of edge detection using the Sobel method. The outputs are captured using 2 OPFs concurrently.

## DoG with PEC in 2nd-Derivative Mode

Difference of Gaussian (DoG) is an edge detection algorithm which approximates the 2nd derivative filtering of an input image. The input image is filtered by two Gaussian kernels with different sigma. The resulting output is subtracted to produce an output similar to the filtering using LoG kernel. This output is fed to PEC which works in the same manner as in LoG method. The post-processing software can use the class/angle information from PEC to build contours for further analysis. The figure below shows the data flow for the DoG method.

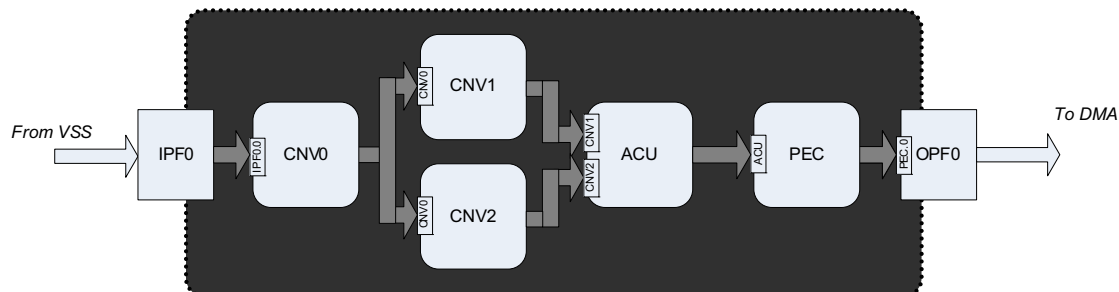


**Figure 30-37:** Difference of Gaussian Data Flow

The input data is first convolved with Gaussian kernel in CNV0. The output is convolved a second time with a Gaussian kernel in CNV1. The resulting output is the result from convolving the original image with a 9x9 Gaussian kernel of sum of sigmas of CNV0 and CNV1. The output from CNV0 can be subtracted

from the output of CNV1 using the ACU. But the two outputs have to be matched for the delay to subtract the pixels at the right instant/location. This is achieved by inserting CNV2 which doesn't modify the data but just matches the delay introduced by CNV1. The kernel for bypassing the input to the output is shown below.

$$Coeff_{CNV2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Figure 30-38:** Bypass Coefficients for CNV2

Note that PEC and OPF settings are the same as in the LoG example.

## Integral of Input Pixels

The IIMx block can be used to compute integral of input pixels as shown in the figure below.

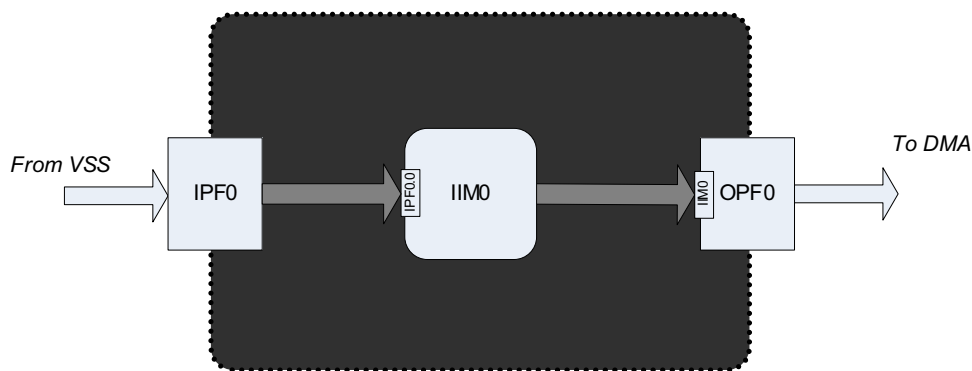


**Figure 30-39:** Integral Image of Input Pixels Data Flow

In camera pipe mode, IIM0 receives data from port 0 of IPF0. The IIM0 can be used to compute Sum-Area-Table (SAT) features by setting `PVP_IIM0_CTL.MODE = 0`. If Rotated-Sum-Area-Table (RSAT) features are required then configure `PVP_IIM0_CTL.MODE = 1`.

Since the integral is computed on single pixels coming in from IPF0, the `PVP_IIM0_CTL.WIDTH = 0` (32-bit input). Note that IPF0 zero extends the incoming pixel values to 32-bits before placing on them on port 0.

In order to reduce the occurrences of overflow/underflow while performing the integral, the IIM supports reducing the input range by shifting the LSBs out of incoming data. The `PVP_IIM0_CTL.SHIFT` bits can be set to a non-zero value to reduce the pixel data range. The IIM can optionally also scale down the output data in horizontal and vertical directions independently. The `PVP_IIM0_SCALE.HSCL` and `PVP_IIM0_SCALE.VSCL` fields can be used for setting the scale factors.

For best operation, set the OPF to receive and transmit 32-bit data using the following bit configuration (PVP\_OPF<sub>x</sub>\_CTL.ISIZE = 0, PVP\_OPF<sub>x</sub>\_CTL.OSIZE = 0).

## Integral of Binary Edge Map

The IIMx block can interface to the THCx to compute integral of edge map as shown in the figure below.

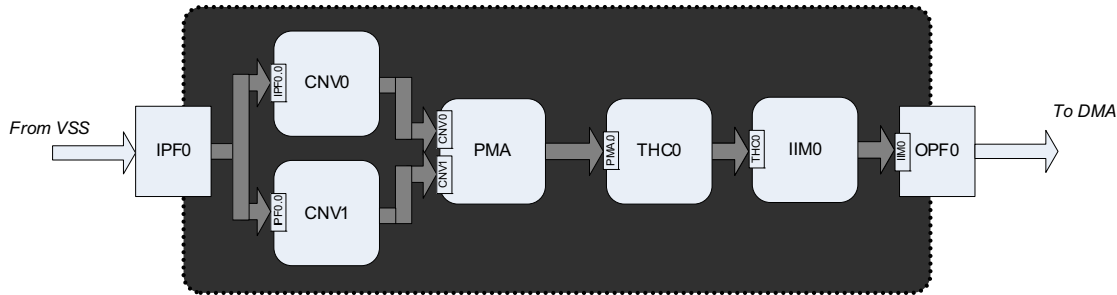


**Figure 30-40:** Integral Image of Binary Edge Map Data Flow

The edge map is generated as described in the Sobel use case example. The 4-bit index output from the THC0 block forms the input to the IIM0 block. By properly choosing THC0 registers (TH0 = 0, TH1 = threshold value and TH2-TH15 = maximum of 16-bit value,  $2^{16}-1$ ) the output from the THC0 block is only 0 and 1, a true binary edge map. The IIM0 block is configured in different modes as explained in the previous example.

## Integral of Variance

The IIMx block can interface to the ACU block to compute integral of squared input values. The input data from IPF0 can be fed to both inputs of the ACU as shown in the figure below.

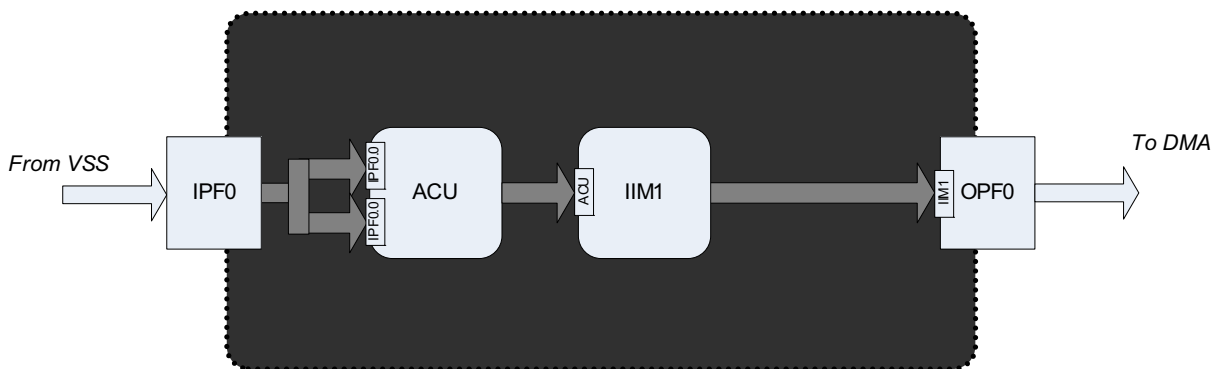


**Figure 30-41:** Integral of Squared Input Values Data Flow

The ACU is set to multiply the input operands (PVP\_ACU\_CTL.SFTINP = 2, PVP\_ACU\_CTL.PRDIINP = 0, PVP\_ACU\_CTL.PRODDP = 0, PVP\_ACU\_SHIFT = 0). The ACU also supports user programmable clipping with values from the PVP\_ACU\_MIN and PVP\_ACU\_MAX registers. The IIM1 block is configured in different modes as explained in the previous example.

## Histogram of Gradients (HoG)

The IIMx block can be used to compute Histogram of Gradients (HoG) features by interfacing to port 1 of the PEC block as shown in the figure below.

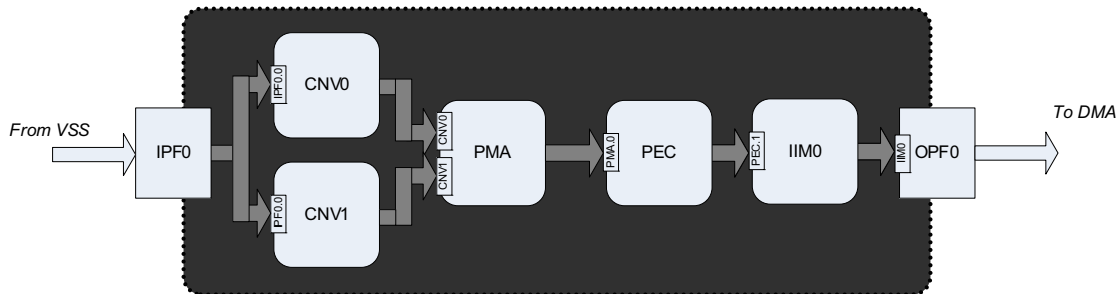


**Figure 30-42:** Histogram of Gradients Data Flow

The PEC block port 1 carries angle information in either 8-bit or 16-bit fields. This format is supported in both modes (1st and 2nd derivative) of the PEC. Port 1 is not valid only when `PVP_PEC_CTL.ZCRSS = 1` in 2nd derivative mode because the angle is not functional.

When the PEC is programmed to output 8-bits per angle bin, (`PVP_PEC_CTL.OSIZE = 0`), there are four angle bins in a 32-bit word output. The IIM0 computes the integral of the individual bins and IIM0 must be set in quad 8-bit mode, (`PVP_IIM0_CTL.WIDTH = 3`).

When the PEC is programmed to output 16-bits per angle bin, (`PVP_PEC_CTL.OSIZE = 1`), there are two angle bins in a 32-bit word output. The IIM0 computes the integral of the individual bins and IIM0 must be set in dual 16-bit mode, (`PVP_IIM0_CTL.WIDTH = 1`).

For optimal operation, set the OPF to receive and transmit 32-bit data (`PVP_OPF0_CTL.ISIZE = 0`, `PVP_OPF0_CTL.OSIZE = 0`).

## Event Control

This section describes PVP event control issues, including interrupt signals, status/error signals, and finish commands.

### Interrupt Signals

Although the PVP features two interrupt outputs, much of the software interaction occurs through the interrupts of the associated DMAs. For example, the PVP interrupt may signal that a frame has been completely received and processed, but only the completion interrupt of the data output DMA can tell whether all results are stored in system memory.

The PVP features a single, 32-bit wide event status register, `PVP_STAT`, as shown in figure below. It contains read-only status bits for camera pipe control, memory pipe control and individual processing blocks. All flags in this register are generated and cleared by hardware.



The PVP\_ILAT register contains the latched counterparts of the status bits in the PVP\_STAT register. If a status bit toggles from 0 to 1 the respective latch bit is set automatically. The latches in the PVP\_ILAT register are cleared by software using the W1C method.

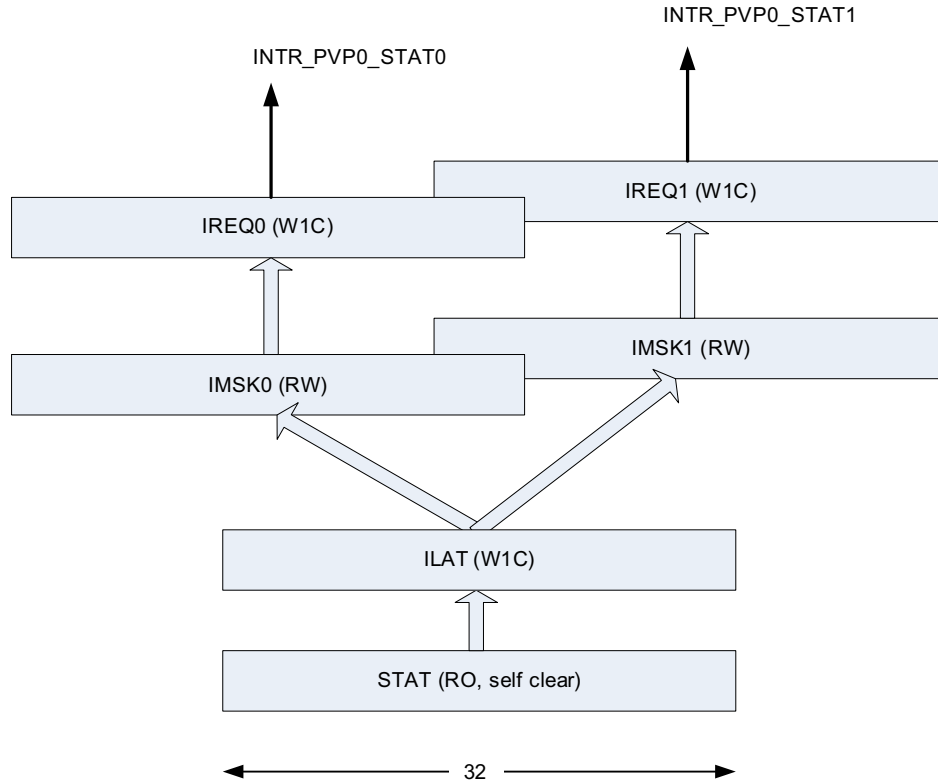


**Figure 30-43:** Interrupt Flow

The two PVP\_IMSKx registers can unmask the interrupt behavior of the latches. If bit 13 in the status register (PVP\_STAT.CPRDY, camera pipe ready) is set by hardware, then latch 13 PVP\_ILAT.CPRDY is also set. By default interrupts are not yet requested. If software sets bit 13 of the PVP\_IMSK0 register, then the CPRDY bit causes an interrupt on the INTR\_PVP\_STAT0 output. If however software sets bit 13 of PVP\_IMSK1, the CPRDY event triggers an interrupt on the INTR\_PVP\_STAT1 output.

On the ADSP-BF609 processor the STAT0 output is connected to system event channel 115 and the STAT1 output is connected to channel 121.

The two PVP\_IREQx registers can be read by software to determine which of the interrupts enabled on a respective channel is pending. Interrupt request bits are a logical AND of mask and latch bits. The interrupt outputs are the logical OR of all interrupt requests in the given PVP\_IREQx register.

Software can acknowledge interrupts by a W1C operation to either the PVP\_IREQx or the PVP\_ILATx registers. Either operation clears the latches in the respective PVP\_ILATx register.

## Status and Error Signals

All PVP status and error events are reported by the `PVP_STAT` register. It is noted that further interrupts and error signals out of PPI, PIXC and DMA channels are highly relevant to PVP operation.

The `PVP_STAT` register contains the following status and error bits for pipe control. There is always one status bit for the camera pipe and one for the memory pipe:

- **Daisy Chain Completion.** The `CPDC` and `MPDC` bits signal whether a configuration update is currently daisy chaining through the pipe. The bits are set along with the `PVP_IPFx_CFG.START` bit and are cleared if all blocks in the pipe have been updated.
- **Write Error.** The `CPWRERR` and `MPWRERR` bits are set when an MMR write is attempted during ongoing daisy chaining of configuration and are cleared at the end of the daisy chain operation.
- **Completion of Drain operation.** The `CPDRN` and `MPDRN` bits get set when the pipe has finished draining in response to `PVP_IPFx_PIECTL.DRAIN` bit. The bits are cleared when `PVP_IPFx_CFG.START` is set next time.
- **Pipe Ready.** The `CPRDY` and `MPRDY` bits indicate that the respective pipe is ready for a new configuration. They are set along with the drain completion bits and cleared by daisy chain completion.

The `PVP_STAT` register contains the following status bit for the data flow.

- **Data Events.** The `CPOPFDAT` and `MPOPFDAT` bits are set when all enabled output formatters have processed the first data word. The bits are cleared with the next data row.

The `PVP_STAT` register contains the following error bits for signaling data transfer errors in camera pipe. Since the memory pipe is designed to not overflow it does not need the following error flags.

- **Output FIFO Overflow.** The `OPF0OVF`, `OPF1OVF` and `OPF2OVF` bits are set when the FIFO overflows, which is fed by OPF outputs and drained by data output DMA. The bits are cleared when the first data value of the next row is processed by the OPF.
- **Status FIFO Overflow.** The `CPSTOVF` flag is similar to `OPFxOVF` bits. It is set when the status output FIFO overflows.
- **Input Overflow.** The `CPIPF0VF` flag is set when data overflows at IPF0. This flag can only be cleared by a camera pipe disable (`PVP_CTL.CPEN = 0`).

The `PVP_STAT` register contains the following status and error bits to report the status of individual processing blocks.

- **Histogram Ready.** The `THCORDY` and `THC1RDY` bits indicate that the THC histogram counters have been updated. Set at the end of a frame when `PVP_IPFx_FCNT = zero` or has counted down to zero and `PVP_IPFx_PIECTL.STATEN = 1`.
- **Integral Unsigned Overflow.** The `IIM0UOVF` and `IIM1UOVF` bits are set when bit 31 of the integral counter in the IIM overflowed during an on-going frame.

- **Integral Signed Overflow.** The IIM0SOVF and IIM1SOVF bits are set when bit 30 of the integral counter in the IIM overflowed during an on-going frame
- **Divide by Zero.** The ACUDIVERR bit is set by the ACU when a divide-by-zero operation was attempted during the on-going frame.
- **ACU Output Saturation.** The ACUOUTSAT bit is set when, during the on-going frame, data was saturated due to the PVP\_ACU\_MIN/PVP\_ACU\_MAX mechanism.
- **ACU Multiplication Saturation.** The ACUPRODSAT bit is set when, during the on-going frame, a multiplication product was saturated.
- **ACU Addition Saturation.** The ACUSUMSAT bit is set when, during the on-going frame, the sum of an addition or subtraction was saturated.

## Finish Commands

Output formatters are the blocks that drive data output DMAs. They use a special DMA feature where they can on demand send a finish command to the DMA channel to signal that the work unit has completed.

The three camera pipe output formatters, OPF0, OPF1 and OPF2 send a finish command when the FIFO overflows. When this occurs, the overflow is reported to the PVP\_STAT register, the FIFO is cleared, and a finish command is sent to the DMA channel at the end of the corrupted frame. The PVP and DMA automatically re synchronize with the next frame boundary. Due to the large depth of the data output FIFO, multiple frames may stick in the FIFO by the time the error occurs. In case of very small frames software may not know whether the automatic recovery dropped one or multiple frames.

The Finish command is also used for signaling. The PVP\_OPF<sub>x</sub>\_CTL.FINISH bit configures this operation. When set, the respective output formatter always issues a Finish command at the end of each frame. This feature is valuable when the exact amount of transfers per frame is not known.

A good example is with run-length compression mode, where the number of output reports generated depends on the incoming data. The DMA work unit is pre configured for a certain number of transfers and only completes if all transfers have taken place. Typically, the run-length compression output DMA is set to the maximum value of reports a frame can generate. This value depends on the PVP\_THC<sub>x</sub>\_RMAXREP register setting. If PVP\_OPF<sub>x</sub>\_CTL.FINISH = 1 the output DMA can terminate before all DMA transfers have occurred. The DMA can immediately progress to the next frame's operation. When using this feature, software needs to be aware that not all memory cells reserved for the operation might be filled with new value.

## Programming Model

This section describes PVP programming techniques, including configuration for pipes, blocks, daisy chains, job lists, and reports.

## Configuring Pipe Structure

Most PVP processing blocks have a single fixed output while others have multiple outputs. The PVP also has one or two inputs that contain multiplexers. This allows system designers to choose what block output the input should be connected to.

The pipeline structure is composed by programming multiplexers from the outputs back to the inputs. By definition, this starts with one of the OPF blocks. For example, if the input of OPF0 is assigned to one of the three output ports of an empty camera pipe on IPF0, a camera pipe without any mathematical mean has been created. If, however, the input of OPF0 is assigned to the output of THC1, and the input of THC1 is assigned to an output of IPF0, a camera pipe with a single mathematical function has been created.

To continue the example, one output of the IPF0 can perform 3-way data broadcasting with the following configuration.

- OPF1 input is connected to the IIM0 output
- The IIM0 input is connected to the ACU output
- The ACU has two inputs and both inputs are connected to the same output of IPF0 and THC1

Similarly, to configure OPF2 to probe the PMA's output use the following configuration.

- Connect the two PMA inputs to CNV1 and CNV2
- Connect these to CNV0's output
- CNV0's input finally receives data from a different port of IPF0

The following diagram illustrates this example. Additionally, it shows the UDS block configured in memory pipe between OPF3 and IPF1.

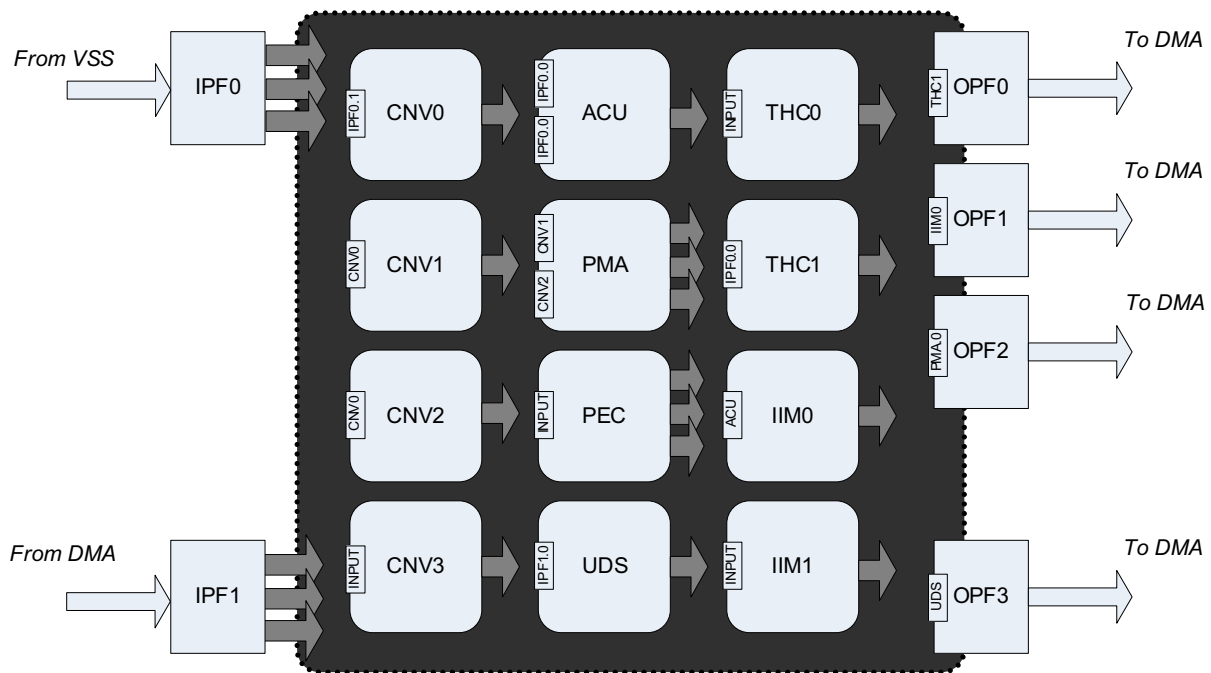


**Figure 30-44:** Pipe Structure Example Configuration

The next figure shows the data flows in the same example configuration. Note the broadcast at the IPF0 and CNV0 outputs. Note also how threads are united by the ACU and PMA blocks.

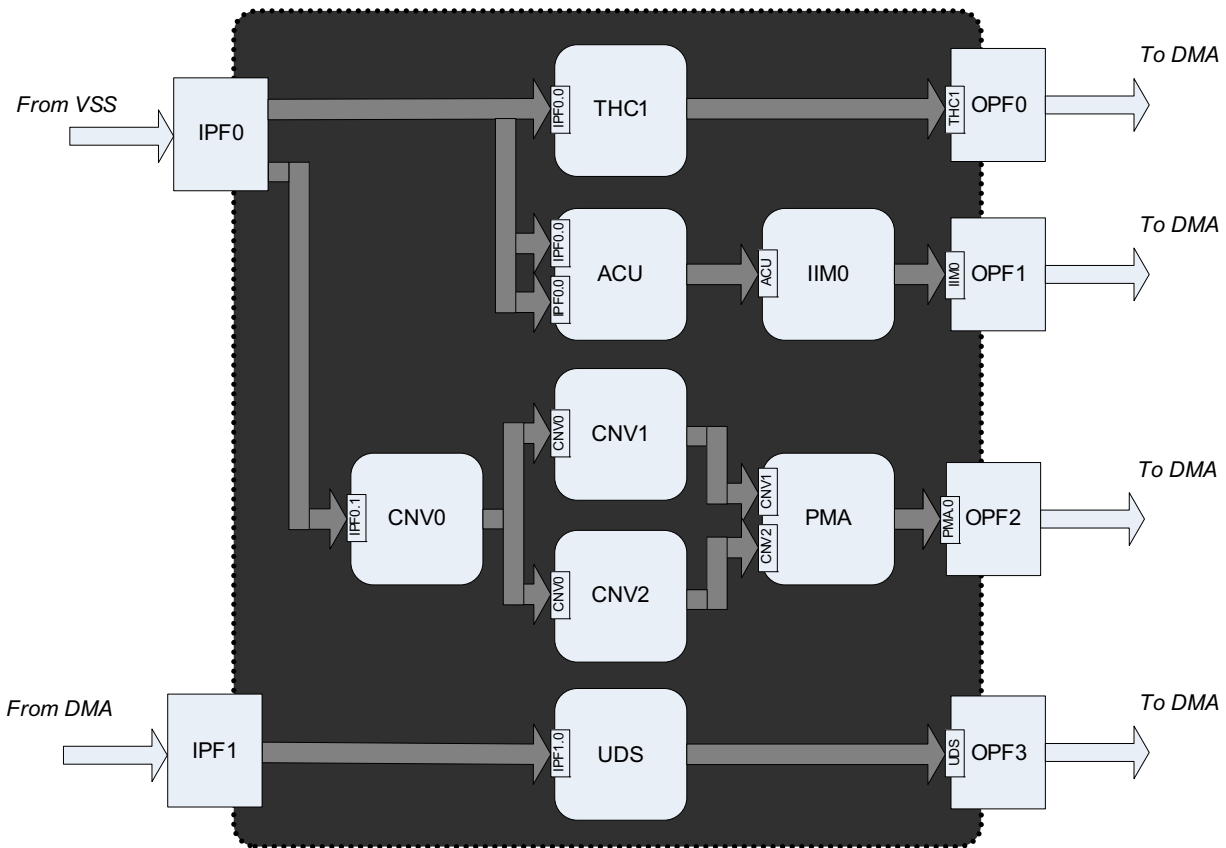


**Figure 30-45:** Data Flow in Example Configuration

Although the example above has significant processing power, it does not use the THC0, IIM1, CNV3 and PEC processing blocks.

To program the input multiplexers, each processing element features a pipe configuration register. The `PVP_xxx_CFG.IBLOCK0` bit field specifies the identifier of the desired source block. The `PVP_xxx_CFG.IPORT0` field is used if the selected source block has multiple outputs. Since the PMA and ACU have two inputs, their configuration register consistently features the `IBLOCK1` and `IPORT1` fields.

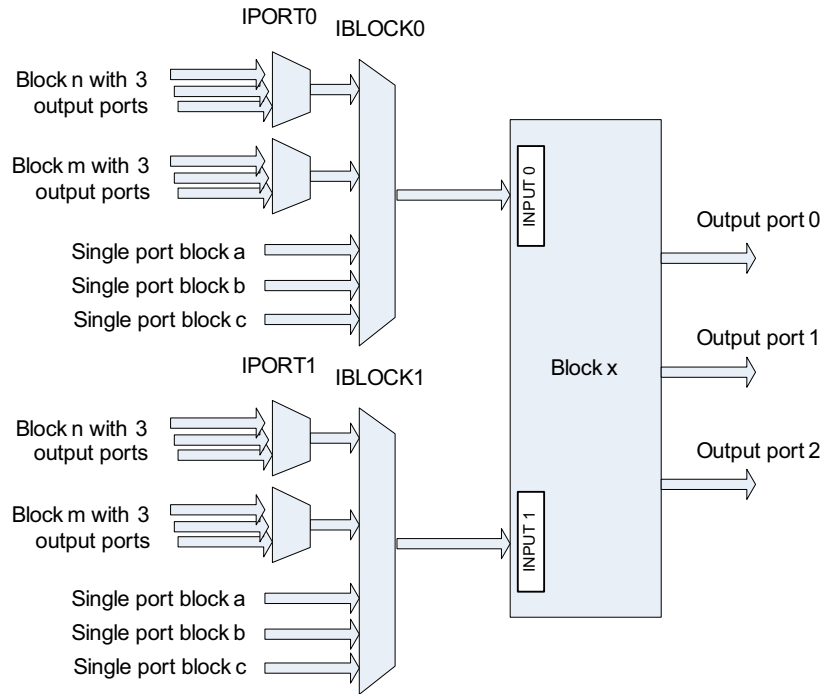


**Figure 30-46:** Input Port Selection

**Table 30-28:** Block IDs

| Block | Block ID | Block | Block ID | Block | Block ID | Block | Block ID |
|-------|----------|-------|----------|-------|----------|-------|----------|
| OPF0  | 0x01     | OPF1  | 0x02     | OPF2  | 0x03     | OPF3  | 0x04     |
| IPF0  | 0x0C     | IPF1  | 0x0E     | THC0  | 0x20     | THC1  | 0x28     |
| CNV0  | 0x10     | CNV1  | 0x14     | CNV2  | 0x18     | CNV3  | 0x1C     |
| ACU   | 0x08     | PMA   | 0x30     | PEC   | 0x05     | UDS   | 0x0A     |
| IIM0  | 0x06     | IIM1  | 0x07     |       |          |       |          |

It is mandatory that all blocks that are used have their `PVP_XXX_CFG.MPIPE` bit properly configured. All blocks that contribute to the OPF3 output, (all blocks operating in the memory pipe), must have the `MPIPE` bit set. For all non contributors, the `MPIPE` bit must be cleared.

**NOTE:** It is user's responsibility to ensure that the `MPIPE` settings match with the assignments of OPF0-2 versus OPF-3. Broadcasts and unions from the camera pipe to the memory pipe and vice versa are absolutely forbidden.

A valid pipe always starts at the input formatter and ends at the output formatter with one exception. Pipes can end at the THC blocks if the data output of the THC blocks is not of interest. Then, the THC blocks

can still operate in histogram mode, where the histogram is outputted to the status DMA. A limitation applies in that such a histogram-only THC setup cannot partner with UDS block in memory pipe mode.

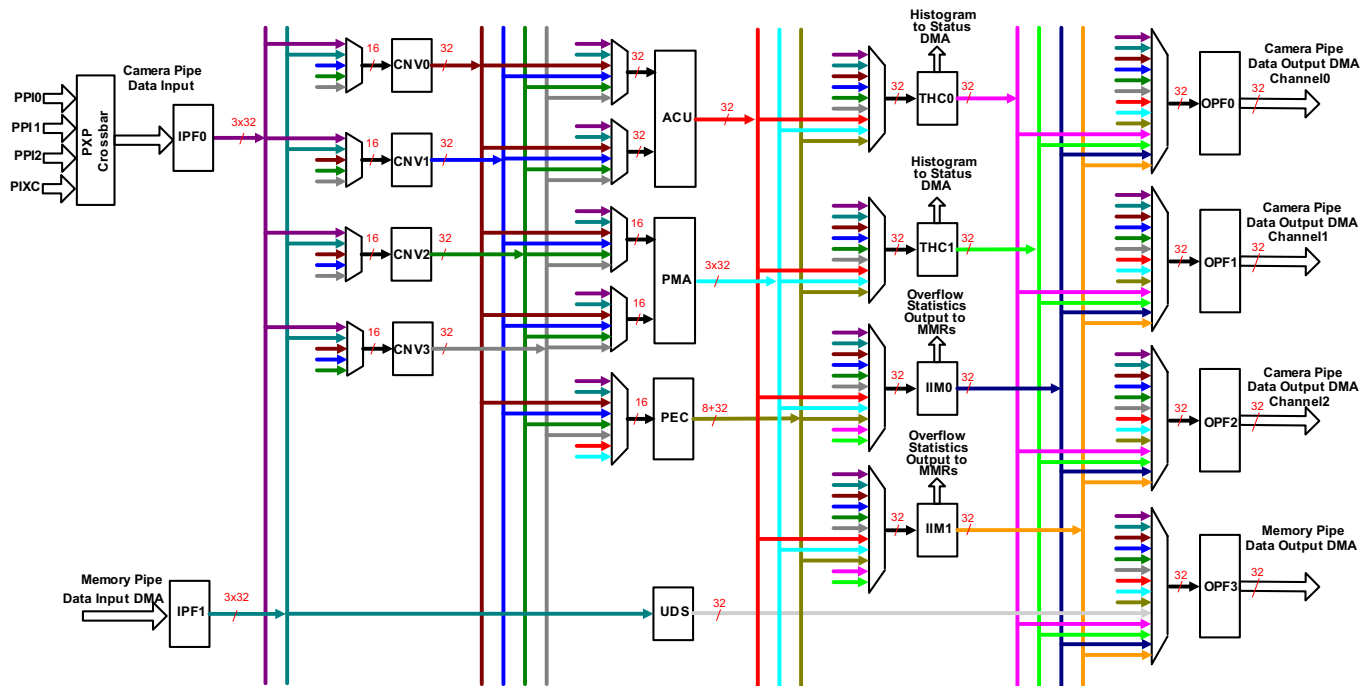


Figure 30-47: Block Connectivity Overview

## Register Based Camera Pipe Configuration

To configure the camera pipe in the static way, use the register-based programming method. In this case all pipe control and processing blocks can be programmed by writing configuration and coefficients into memory-mapped registers (MMRs). Typically, the following procedure is used.

1. Disable all involved PVP blocks, all involved VSS elements and all involved DMA channels to ensure that all data FIFOs are empty
2. Configure video interconnect routing
3. Configure pin muxing for PPI input operation
4. Set the `PVP_CTL.PVPEN` and `PVP_CTL.CPEN` bits to enable camera pipe mode
5. Configure and enable all output DMAs using either the register-based or the descriptor-based method.
6. Configure all camera pipe processing blocks starting from the output formatters to IPF0. Ensure that the structure of the camera pipes are configured using the `PVP_XXX_CFG` registers and ensure that the appropriate `MPIPE` bits are cleared and the appropriate `START` bits are set.



7. Enable the IPF0 block. If continued camera pipe operation is desired, set `PVP_IPF0_FCNT = 0`. Finally, setting the `PVP_IPF0_CFG.START` bit enables the entire camera pipe.
8. Configure PPI DMAs
9. Enable the PPI (and optionally the PIXC)

Once all blocks are enabled the PVP operation starts with the next vertical frame sync (VSYNC). There is some flexibility in the order of above procedure. For example, the PVP camera pipe can be reconfigured while the PPI is kept enabled. Then, the camera pipe starts operating with the next VSYNC. If frame characteristics change however, (for example as in the case of the PPI's window feature), care is required to ensure that the PVP output (2-dimensional) DMAs match with the window settings every frame.

All block registers are double buffered. All coefficient registers can be written any time and are properly synchronized with pipe progress by hardware. However, the `PVP_XXX_CFG` registers must only be written when a pipe re-configuration is desired for the next set of frames. Writes to the `PVP_IPFx_CFG` registers need to be synchronized with ongoing activity.

**NOTE:** Register writes during daisy chain load period are not allowed. Writes are ignored and error bits are set. MMR reads incur 1 cycle of latency while writes incur no latency. Also note that access to holds/read only gives MMR error.

## Block Configuration Structure

The PVP can be programmed by writing to memory-mapped registers directly. However, an alternative configuration method using DMA channels can be used. This method is derived from the descriptor-based programming method of the DMA controllers themselves and works well with DMA programming.

The PVP features two configuration DMA inputs, one for camera pipe and one for memory pipe operation. These DMAs enable on-the-fly re-configuration of the two pipes independently. Each pipe can be configured or re-configured according to timing needs without core interaction.

The configuration DMA channels write to the MMR register space. For the processing blocks it does not make any difference whether a memory-mapped register has been written by a core write or by configuration DMA. Certainly, unlike core writes, configuration DMA's write timing is ensured by hardware and not subject of interrupt latencies.

The block configuration structure (BCS) is a telegram loaded by either of the configuration DMAs. It consists of a 32-bit block configuration header (BCH) which is followed by a number of 32-bit configuration words that target memory-mapped registers.

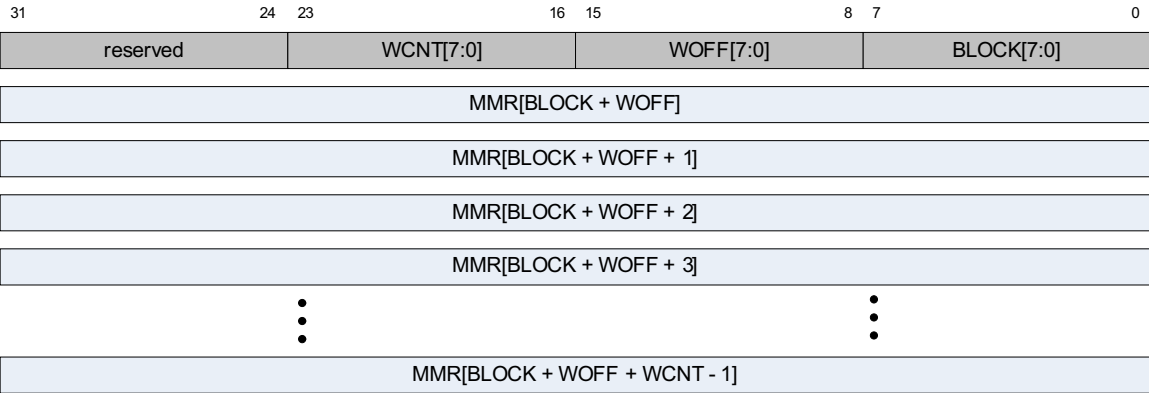


Figure 30-48: Block Configuration Structure

When the configuration DMA fetches a BCS, the BCH specifies the MMR destination of the subsequent configuration words. The 8-bit BCH.BLOCK field specifies which PVP processing block the subsequent configuration words belong to. This uses the same block id scheme as in. The 8-bit BCH.WOFF field specifies the word offset versus that first register of the block's register space. The 8-bit BCH.WCNT field specifies how many registers are to be written to the specified register space. The BCH.RESERVED field must be filled with zeros.

Configuration DMAs not only can fetch single BCSs, but also can fetch a series of BCSs at once. A series of BCHs is called block configuration list (BCL) and is illustrated in the following figure.

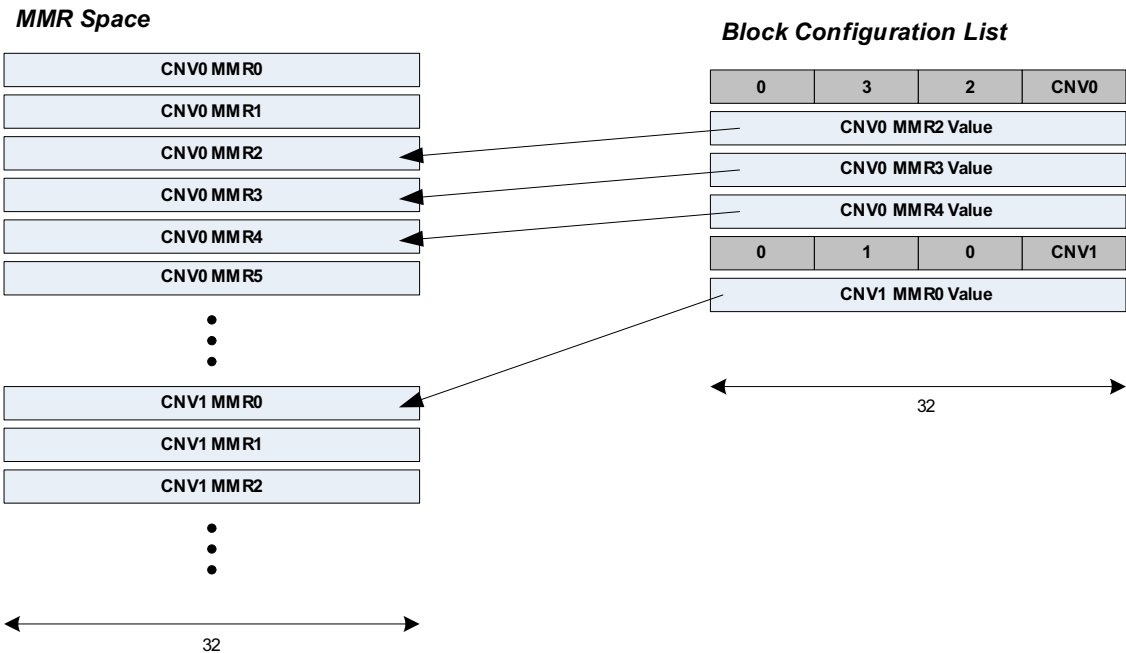


Figure 30-49: Block Configuration List Example

The example block configuration list consists of two BCSs. The first targets processing element CNV0. It contains three 32-bit configuration values to be written into three MMR registers that are contiguous in

register space. The register space contains multiple registers dedicated to the CNV0 block. Since the example BCS does not target the first three MMRs in CNV0 space, but rather the three from MMR2 on, the `BCH.WOFF` field is set to a value of 2. The second BCS target is CNV1. It only writes to a single register and since it is the first register in CNV1 space the `BCH.WOFF` field = 0.

## Initial Configuration Fetch

After the camera pipe or memory pipe have been enabled by the `PVP_CTL.PVPEN` or `PVP_CTL.CPEN` bits, the respective input formatter immediately requests a BCL fetch from the configuration DMA channel. If the configuration DMA does not grant the request because it is either not ready or not enabled, the pipe engine stalls until either the pipe is configured by MMR writes or the DMA starts granting. The IPFs do not accept any data on their data input until the pipe is fully configured and enabled.

When the configuration DMA is granted, the input formatters fetch BCL words until the `PVP_IPFx_CFG.START` bit is set. The BCL is assumed to describe a valid pipe configuration and to write a 1 to the `PVP_XXX_CFG.START` bits of all involved blocks. The `START` bit can be seen as a self-clearing block enable bit. The self-clearing nature of this bit ensures that software does not need to perform garbage collection at or after pipe re-configuration. Blocks that are no longer used are automatically disabled.

The `START` bits of the input formatters have additional purpose also enable the entire camera or memory pipe. Therefore, while the order of BCSs inside a BCL does not matter, only the very last BCS writes into the `PVP_IPF1_CFG` register.

Once the `PVP_IPFx_CFG.START` bit has been written, the IPF0 starts accepting data VSS, or IPF1 starts requesting data from data input DMA and the pipe starts processing.

## Descriptor-based Memory Pipe Configuration

To enable memory pipe operation, set the `PVP_CTL.PVPEN` and `PVP_CTL.MPEN` bits. Additionally, the `PVP_CTL.CLKDIV` must be configured properly. While the camera pipe is clocked by the data clock of the receiving PPI, the clock for the memory pipe is derived from system clock `SCLK0`. The PVP is designed for maximum operating frequency of 83MHz. For best performance:

- Clear `PVP_CTL.CLKDIV` if `SCLK0` <= 83.3 MHz, so the memory pipe clock equals `SCLK0`
- Set `PVP_CTL.CLKDIV` if `SCLK0` > 83.3 MHz, so memory pipe clock equals `SCLK0/2`

**NOTE:** Over clocking of memory pipe can lead to unpredictable behavior and thermal silicon defects.

Before enabling a memory pipe job, configure the respective data input and data output DMAs and enabled them immediately. Typically, the completion interrupt of the data output DMA is also enabled. It signals when the PVP memory pipe has completed the job and ensures that the results have been written back to memory.

A BCL structure is expected somewhere in system memory (L1, L2 or L3). This BCL structure can be statically pre configured or dynamically composed at run-time.

Finally, the configuration DMA is configured. The `DMAx_STARTADDR` registers point to the base address of the BCL. As soon as the configuration DMA is enabled the operation takes place autonomously. IPF1 starts requesting data from the data input DMA as soon as all BCL has been fetched, (once the `PVP_IPF1_CFG.START` bit has been set). Software is alerted by the completion interrupt of the data output DMA connected to OPF3.

## Dynamic Pipe Control

To control how a pipe behaves after it has been initially configured, the input formatters use the frame counter (`PVP_IPFx_FCNT`) registers. If these registers = 0, the respective pipe repeatedly operates in the same mode as long as it is not disabled by the `PVP_CTL.CPEN` or `PVP_CTL.MPEN` bit. If `FCNT = 1`, the pipe operates on exactly one frame and then stalls. If `FCNT` is set to a value of `N`, exactly `N` frames are processed.

If `FCNT = 0` or `FCNT = 1` the input formatters request a new BCL from configuration DMA every frame. If `FCNT > 1`, a new BCL is only requested every `N`th frame. Similarly, status reports can only be transmitted at `FCNT` frame boundaries.

Since the PVP is a pipelined engine, it takes `X + D` data clocks to process `X` data words, where `D` stands for the depth of the pipeline. The PVP is clocked by data and therefore, after `N` frames have been processed, a number of results (`D`) still stick in the pipe and do not yet progress to the data output DMA channel because the pipe is waiting for further BCL instructions. The first data clocks of the future operation then implicitly clock the older results out.

To automatically drain the remaining result words, set `DRAIN = 1`. This configuration allows IPF0 to pass further `D` data clocks into the pipe for this purpose. Keep Upstream PPI, PIXC and VSS (and video source device) enabled so they can provide the required clocks. However, IPF1 cannot rely on receiving further data clocks from data input DMA. Rather, if `DRAIN = 1`, IPF1 generates the required clocks itself artificially.

The `FCNT` and `DRAIN` bits partner in instructing the input formatter on how to control the pipe as shown in the following table.

**Table 30-29:** Operating Modes as by `FCNT` and `DRAIN`

| FCNT | DRAIN | Mode                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0    | 0     | <p>Mode 0: Continuous Frame Mode</p> <p>Frames are continuously requested by the IPFx and feed the PVP pipe. The Configuration DMA is constantly requesting BCL words every frame. If new Configuration is granted by DMA or by MMR writes, it is applied on the next frame boundary.</p> <p>Processing Blocks do not auto-disable. New Configuration must not set <code>PVP_xxx_CFG.START</code> bit of any block again.</p> |

**Table 30-29:** Operating Modes as by FCNT and DRAIN (Continued)

| FCNT | DRAIN | Mode                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| >0   | 0     | <p>Mode 1: Back-to-Back Mode</p> <p>FCNT frames are requested by the IPFx and these feed the pipe. The last data words will be stuck in the pipe. The pipe stalls, and IPF x is requesting new instructions from Configuration DMA.</p> <p>If new Configuration is granted by DMA or by MMR writes, it is applied for the next FCNT set of frames. The first data clocks of the new set push the remaining words of the former set out to the data output DMAs.</p> <p>Only IPF blocks auto-disable, other Processing Blocks do not. New Configuration needs to reset <b>START</b> bit of IPFx, but must not set the <b>START</b> bit of the other blocks. Ideally, writes to <code>PVP_xxx_CFG</code> registers of other blocks than IPFx are avoided altogether.</p> |
| >0   | 1     | <p>Mode 2: Autocompletion Mode</p> <p>FCNT frames are requested by the IPFx and these feed the pipe. The last data words are automatically pushed out and the operation automatically completes.</p> <p>If new Configuration is granted by DMA or by MMR writes, it is applied for the next FCNT set of frames.</p> <p>All Processing Blocks auto-disable. New Configuration must set all wanted <b>START</b> bits again.</p>                                                                                                                                                                                                                                                                                                                                          |
| 0    | 1     | <p>Mode 3: Drain Now Instruction</p> <p>This combination instructs the IPF to flush all content out of the pipe.</p> <p>All Processing Blocks auto-disable. New Configuration must set all wanted <b>START</b> bits again.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

Mode 0 is popular for camera pipe operation. Often, the camera pipe is enabled statically and updates to the coefficients is only required from time to time. On demand software may enable the configuration DMA in stop mode to trigger the fetch of one single BCL. In this mode the `PVP_IPFx_TAG` register is useful to match status results with a respective configuration. The new BCL only updates coefficients of processing blocks, it does not alter the pipe configuration.

Mode 1 is useful in camera pipe mode when coefficients have to change every Nth frame. It supports on-the-fly reprogramming of coefficients. However, it does not support on-the-fly reprogramming of pipe configurations. The following settings must not change unless the **DRAIN** bit is set:

- Pipe structure (`PVP_xxx_CFG` registers)
- Fundamental operating modes (like 1st derivative to 2nd derivative mode of PEC)
- Input format and OPORT configuration (`PVP_IPFx_CTL` registers)
- Horizontal size of input frame (**HCNT**)

If any of the above settings change on-the-fly, then the `DRAIN` bit must be set for proper operation. Mode 2 is used. In Memory Pipe mode the `DRAIN` bit causes of few clock cycles of overhead depending to pipeline depth. In camera pipe mode the `DRAIN` bit causes the loss of an entire frame. Operation stops after the pipe has been drained and resumes with the next incoming `VSYNC` signal.

Whenever the next PVP job stalls until the completion event of the data output DMA is flagged either as an interrupt or as a system trigger, setting the `DRAIN` bit is a requirement at the application level. The data output DMA work unit does not complete as long as related data sticks in the pipe. The data in the pipe does not progress until the configuration DMA and data source grant. Such deadlock situations can be avoided by using mode 2. Once a deadlock occurs, it can be resolved by mode 3 operation.

Mode 3 is more like a one-time instruction than an operating mode. It is used whenever software needs to respond to unpredicted events. For example, while the camera pipe is operating in mode 1, software notices that the future frames need to have alternating `HCNT` settings. If the pipe is already stalled waiting for new `BCL` instructions, a `BCL` containing a Drain Now instruction can gracefully switch the pipe from mode 1 into mode 2. All pending results in the pipe are still properly shifted out before the pipe resumes operation with the new `HCNT` settings after the subsequent `BCL` fetch.

The following list describes the difference between the modes in disabling the processing blocks after operation.

- In mode 1 the hardware only disables the input formatters after `FCNT` expires. All other blocks remain enabled. In principle, only the `PVP_IPFx_CFG` register needs to be rewritten to set the `START` bit again to initiate another operation. New configurations must not set the `START` bit of any other block another time. If a new write to the `PVP_XXX_CFG` register cannot be avoided, the `START` bit shall be written by a zero value.
- In mode 2 and mode 3 all blocks are disabled by hardware after the drain operation. For future operation rewrote the `PVP_XXX_CFG` register of all included blocks to set the `START` bit again.
- In mode 0 new configurations must not set the `START` bit of any block another time, not even the one of the input formatters. If a new write to the `PVP_XXX_CFG` register cannot be avoided, write the `START` bit to a zero value.

The `PVP_STAT` and `PVP_ILAT` registers report when the camera pipes (`CPDRN` bit) and the memory pipe (`MPDRN` bit) are drained. The ready bits, `PVP_STAT.CPRDY` and `PVP_STAT.MPRDY`, report whether `PVP_IPF0_CFG.START` and `PVP_IPF1_CFG.START` bits are ready to be set.

## Latency and Pipeline Depth

Most of PVP processing blocks have internal data buffers, if not row buffers. Before a first result can be generated, these buffers need to be filled with valid input data, so that the mathematical operation can be performed. The amount of buffers needed for the operation defines the latency that a block is adding to the chain if inserted into a pipe. This equals the number of artificial clocks that need to be applied in the case of a drain operation. Due to their 2-dimensional nature the `CNVx`, `PEC` and `UDS` blocks have local row buffers where they intermediately store one or multiple data rows. Consequently, their latency is much higher than that of the `IPFx`, `ACU`, `PMA` or `OPF` blocks. The `IIMx` block is special in that although it

performs 2-dimensional operations as well, it can generate outputs early, because results do not depend on future data.

**Table 30-30:** Block Latencies

| Processing Block | Latency [Data Clocks]                                                                 |
|------------------|---------------------------------------------------------------------------------------|
| IPFx             | 5                                                                                     |
| OPFx             | 1                                                                                     |
| THCx             | 3                                                                                     |
| CNVx             | 2x HCNT + 14                                                                          |
| PMA              | 5                                                                                     |
| ACU              | 7                                                                                     |
| PEC              | 2x HCNT + 11 if 2 <sup>nd</sup> derivative mode with ZCRSS=0,<br>otherwise 1xHCNT + 6 |
| IIMx             | 6                                                                                     |
| UDS              | varies with settings                                                                  |

## Daisy Chain Configuration

Configuration and coefficient MMRs of all processing blocks are double buffered. Whenever `FCNT` is zero and the `PVP_IPFx:CFG.START` bit = 1, the values are copied from MMR registers into application buffers at the frame boundary. Due to the pipelined nature of the PVP not all values are copied at the same time. Rather, the timing is closely related to how the `VSYNC` of the next frame progresses through the pipe. New settings apply to the first pixel of the new frame immediately after the last pixel of the old frame has been processed. The update command is daisy chained through the pipe, starting from the IPFx down to the OPFs.

While sequencing through the daisy chain operation, hardware clears the `PVP_xxx_CFG.START` bits of all involved processing blocks in MMR space. Write conflicts can occur if software attempts to write the `PVP_xxx_CFG` registers during the daisy chain.

The `PVP_STAT` register provides two status bits that report whether a daisy chain operation is ongoing in the camera pipes (`PVP_STAT.CPDC`) or in the memory pipe (`PVP_STAT.MPDC`). These bits are set along with the `PVP_IPFx_CFG.START` bit and are cleared if the output formatters have been updated.

In camera pipe mode the `CPDC` bit is cleared only after the daisy chain progresses through the furthest of the enabled OPFs. During this time, software should not write into PVP MMR space. If software mistak-

only does, this is reported by the `CPWRERR` and `MPWRERR` flags in the same register. While all status bits in the `PVP_STAT` register are self clearing, the `PVP_ILAT` register latches the events until cleared by a software handshake.

A block which is enabled in a given pipe, either camera or memory, can be moved to the other pipe only after the drain done command is received for the former pipe. This ensures that the module completely processes all the pixels in one pipe, and then is disabled before being re-enabled for pixel processing in the next pipe.

**NOTE:** Camera and memory pipes can be configured by MMR writes or BCL fetches. These are separate control mechanisms and are not intended to be mixed for a given pipe during ongoing operation. Before switching from one method to the other inspect the daisy chain and drain status bits in the `PVP_STAT` register to ensure no operation from the other method is pending.

## Job Lists

This section provides information on how to maintain lists of multiple operations for sequential processing (co-processor like mid-processing) by the PVP memory pipe (PVP memory pipe job list). Sequential processing requires minimal overhead for the core and has high PVP performance. The information this section also applies to the camera pipe.

In this section, data to be processed within the PVP memory pipe is assumed to be completely stored in memory. This is in contrast to the camera pipes in when processing data streams on the fly gated by an external (PIXEL) clock. In that case the processing data is not fully loaded into memory and additional measures are required to control the input data DMA so as not to run out of valid data. While this is technically possible it is not part of the following description.

This section requires basic knowledge of the DMA functionality, especially the use of linked list descriptors as well as 1-dimensional and 2-dimensional DMA operations. For more information, see the DMA and Trigger Routing chapters.

To sequentially schedule multiple PVP memory pipe operations, correctly initiate and trigger the following data flows (each handled by a different DMA move engine).

- DMA 45 moves the PVP memory pipe configuration (BCS block control structure list) of a specific PVP memory pipe job from memory into the PVP, configuring the processing functionality of the elements building the PVP memory pipe.
- DMA 43 moves the data set to be processed from memory into IPF1 using this specific PVP memory pipe configuration.
- DMA 42 moves the results of the PVP memory pipe operation (OPF3) to a destination, usually to memory.

The PVP can optionally generate an additional status (report) data stream originating from IPF1, THC0, or THC1. This data stream also needs to be moved from the PVP memory pipe status output to a destination, (usually to memory) using DMA 44.



To initiate the PVP memory pipe job list linked list, set up three DMA descriptors. Each linked descriptor stores the DMA configuration parameters within memory to maintain the DMA sequences to handle the three data flows, BCS, data IN, data OUT, and optionally status data OUT. If the descriptors are all completed by the core, multiple PVP memory pipe jobs can be chained without any further core intervention as the DMAs run through the chained lists and handle the multiple data flows autonomously. The following description assumes the memory pipe is inactive, (a non initialized state `PVP_CTL.MPEN = 0`).

## Configuration Descriptor List (Static Job List)

To run a PVP memory pipe job from the PVP memory pipe job list, the PVP first loads the memory pipe configuration into the respective shadow registers of the respective PCP elements to build the PVP memory pipe. The configuration within the shadow register is activated (transferred from shadow register to active register) using hardware triggers (staggered transfer). This configuration stream into the shadow registers is handled by DMA 45 which is controlled by a linked descriptor list. The last valid descriptor must contain a `DMA_CFG` field with `FLOW = 0` (STOP Mode) to gracefully stop the DMA 45 after completing the PVP memory pipe job list and avoid loading non valid data into PVP memory pipe configuration register. All `DMA_CFG` descriptor fields within the linked list have `FLOW = 6` or `FLOW = 7` (descriptor list mode) or alternatively `FLOW = 4` (descriptor array mode).

A 1-dimensional DMA operation is shown in the following figure. Note however, a 2-dimensional DMA operation can also be used.

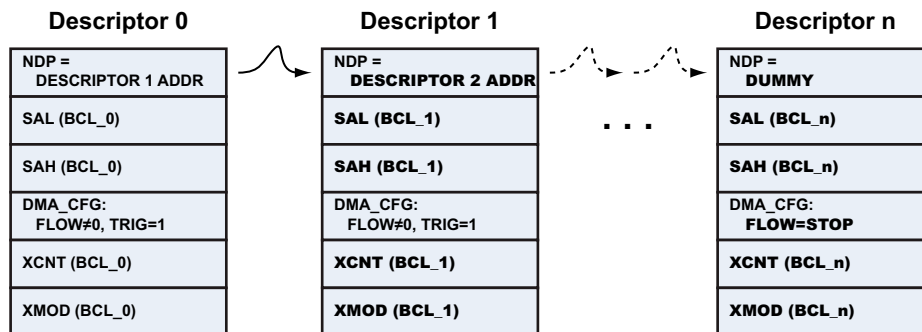


**Figure 30-50:** Static Linked Descriptor List for Configuration Data (BCL Data)

## Global Trigger

After the first initialization of the linked descriptor list the parameter registers of DMA 45 have to be initialized, particularly the `DMA45_DSCPTR_NXT` and `DMA45_CFG` register. The DMA 45 is started but then stalls until the PVP memory pipe is enabled and the `IPF1_CFG` register bit field `FCNT` is equal to 0 (`FCNT = 0`).

To generate a unique synchronization signal that synchronizes the Data Out (DMA 42) and Data In (DMA 43) stream, all `DMA_CFG` descriptor fields of DMA 45 should have the bit field `TRIG` set (`TRIG = 1`). This generates a trigger signal after a new configuration is loaded into the shadow registers.

## Configuration Descriptor List (Dynamic Job List)

When the memory pipe job list is started before the creation of the PVP memory pipe job list is completed by the core (dynamic creation of PVP memory pipe jobs) the following issues must be considered.

- Use mode 1 (back-to-back mode) is for maximum performance. But because this mode moves data of a new job into the PVP memory pipe, while the data of the previous job has not been completely moved out of the PVP memory pipe due to memory pipe latency and the output buffer within OPF3, there must not be any data dependency between input and output data (no recursive data processing). Further the PVP memory pipe structure must not be modified, because the staggered loading of the configuration may not work correctly. A further challenge of this mode is the fact, that the last job cannot be completed, as it requires “dummy” data to be pushed explicitly into the PVP Memory Pipe to shift out the last results.
- Alternatively, dummy data (bubbles) may be implicitly (automatically) appended to every data set to entirely empty the PVP memory pipe before the new data are processed using mode 2 (auto-completion mode) of IPF1. Due to the bubbles moving through the PVP memory pipe, this mode decreases the performance of the PVP, but allows fully interlocked pipelined processing without the need for special care of control conflicts. Mixing both modes is usually not possible, because it requires dynamic modifications of the IPF1 configuration and DMA trigger configuration.

**NOTE:** Special care has to be exercised when using back-to-back mode if the PVP memory pipe job list has started before the creation of the PVP memory pipe job list has been completed by the core, or the data to be processed within the PVP memory pipe job list is not yet available in memory. Auto-completion mode is the preferable mode in these cases.

**NOTE:** Special care has to be exercised when the BCS includes PVP\_XXX\_CFG register updates, (with exception of PVP\_IPF1\_CFG). This requires processing the PVP memory pipe job list in mode 2 (auto-completion mode) as the staggered configuration load of the pipeline elements may no longer work correctly because the position of an element within the pipeline may change and therefore a different staggering order may be required

## Control Data Flow (Dynamic Job List, Auto-completion Mode and Back-to-Back Mode)

Support for dynamic job lists requires that the last valid descriptor (descriptor n) must contain a DMA\_CFG field with TWAIT = 1 (halt and wait for incoming trigger) instead of FLOW = 0 (stop mode) in order to pause and not stop the DMA channel and avoid loading non valid data (job creation not finalized) into the PVP memory pipe configuration register. An additional last descriptor (descriptor m) is reserved in memory, but only contains dummy values and a STOP mode in the DMA\_CFG field to handle erroneous descriptor overrun and enable dynamic memory allocation.

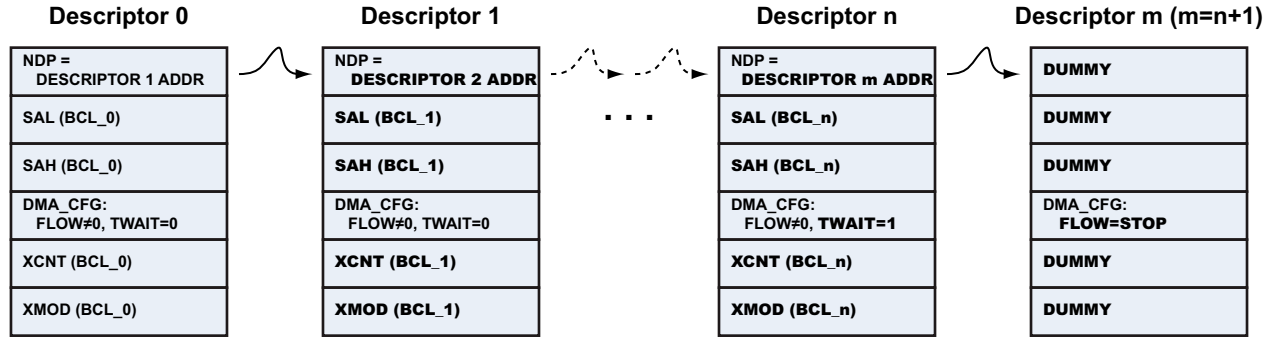


**Figure 30-51:** Linked Descriptor List for Configuration Data (BCL Data) Enabling Dynamic Extension

If adding an additional job to an existing PVP memory pipe job, this new job descriptor replaces the DUMMY descriptor (descriptor m) in memory. This updated descriptor m must now link to a new DUMMY descriptor (descriptor j).

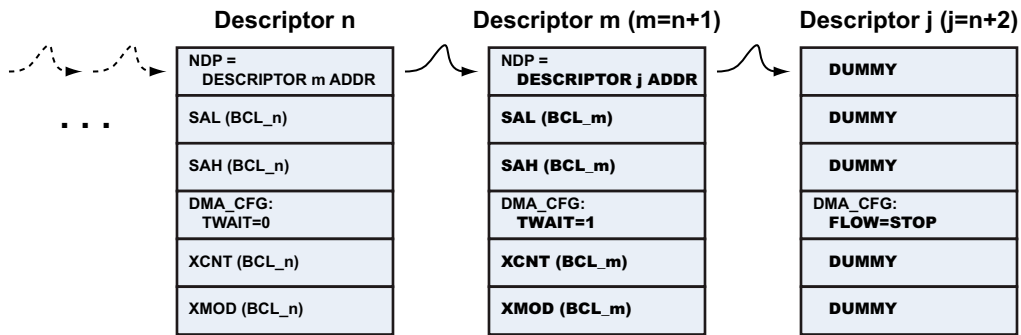


**Figure 30-52:** Dynamic Extending Linked Descriptor List for Configuration Data (Job List)

After completing the descriptor m and the job list extension that handles Data In and Data Out, modify (clear) the `TWAIT` field of the `DMA_CFG` field of descriptor n. To grant a seamless DMA operation where:

- The DMA has already loaded the unmodified descriptor n:
- and has already copied the `TWAIT = 1` descriptor (original descriptor n) into the `DMA45_CFG` register:

Generate a software generated trigger for DMA 45 (and DMA 43 if in back-to-back mode). Therefore if `TWAIT = 1`, a trigger is generated within the trigger routing unit (TRU) by software.

Additionally, to grant a seamless DMA operation using auto-completion mode where:

- DMA 42 has already loaded the unmodified descriptor n:
- and therefore already copied the `TWAIT = 1` descriptor (descriptor n) into the `DMA42_CFG` register:

Generate a software generated trigger for DMA 43. Therefore if `TWAIT = 1`, a trigger is generated within the TRU by software.

Take special care if writing the `TWAIT` field of the `DMA_CFG` field and if reading the `DMA45_CFG` and `DMA42_CFG` registers takes longer than the DMA operation loading the configuration data.

**NOTE:** Alternatively, instead of setting `TWAIT = 1`, use `FLOW = 0` as a semaphore (STOP MODE). Set the `DMA45_CFG` register (write operations) to descriptor mode if the DMA stopped.

## Data Out Flow (Static Job List)

After successfully setting up and starting the DMA 45 linked descriptor list for the configuration data of the PVP memory pipe, set up an additional DMA (DMA 42) to move Data Out (result data) from the PVP memory pipe (out of OPF3) into memory. A second linked descriptor list that handles the output data (result) is set up according to the previous description of the `DMA_45` linked description list. Optionally set up DMA 44 using an additional linked descriptor list that handles Status Data Out.

These data streams, the Data Out and the Status Data Out, are handled by dedicated DMAs. DMA 42 handles the Data Out stream and DMA 44 the Status Data Out stream.

Both are controlled by their respective linked descriptor list. The last valid descriptor of each linked list must contain a `DMA_CFG` field having a `FLOW = 0` (STOP Mode) to gracefully stop the DMA 42 and (optionally) DMA 45 after completing the PVP memory pipe job list and to avoid loading non valid data into PVP memory pipe configuration register. All `DMA_CFG` descriptor fields within the linked list have `FLOW = 6` or `FLOW = 7` (descriptor mode) or alternatively `FLOW = 4` (descriptor array mode). Configure the TRU to route the trigger of DMA 45 to the input trigger of DMA 42 and DMA 44.

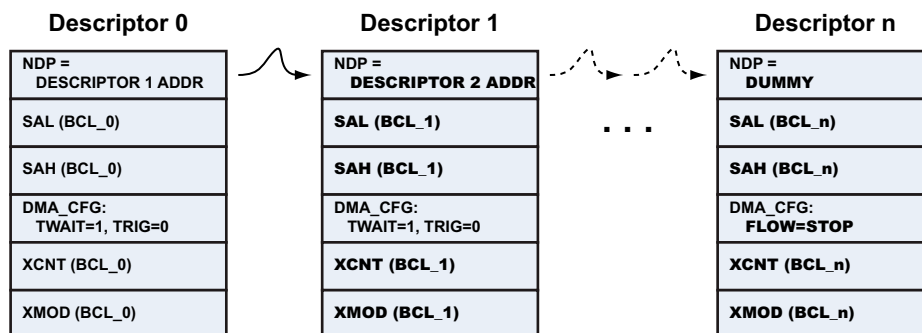


**Figure 30-53:** Static Linked Descriptor List for Data OUT (Result Data) and Optionally Status Data Out

## Data Out Flow (Dynamic Job List)

The following situations describe handling Data Out data streams using back-to-back mode or auto-completion mode. Mixing modes is not possible because it requires modifications of the IPF1 configuration and the DMA trigger configuration.

- Use mode 1 (back-to-back mode) for maximum performance. Note that this mode moves data to be processed by a new job into the IPF1 while the resulting data processed by the previous job has not been completely moved out of the PVP memory pipe. This is because there may not be any data dependency between the input and output data (recursive data processing) in the output buffer within OPF3, as well as PVP memory pipe latency. Further, the PVP memory pipe structure may not be modified because the staggered loading of the configuration may not work correctly. A further challenge of this mode is

it requires “dummy” data to be pushed explicitly into the PVP memory pipe to shift out the last results and therefore the last job cannot be completed.

- Use mode 2 (auto-completion mode) for fully interlocked pipelined processing without care for data hazards (read after write hazard). This mode appends dummy data (bubbles) implicitly (automatically) to every data set to empty the PVP memory pipe before the new data is processed in IPF1. Because the bubbles move through the PVP memory pipe, this mode decreases the performance of the PVP.

Special care must be exercised when using back-to-back mode if the PVP memory pipe job list has been started before the creation of the PVP memory pipe job list has been completed by the core, or if the data to be processed within the PVP memory pipe job list is not yet available in memory. Auto-completion mode should be used in these cases.

## Data Out Flow (Dynamic Job List, Back-to-back Mode)

Dynamic job lists require that the last valid descriptor (descriptor n) for DMA 42 (and optionally DMA 44) to contain a `DMA_CFG` field with `TWAIT = 1` (halt and wait for incoming trigger). This is so the DMA channel pauses instead of stops, and avoids loading non valid data (job creation not finalized) into the PVP memory pipe configuration register. An additional last descriptor (descriptor m) is reserved in memory which contains dummy values and a `FLOW = 0` (STOP mode) in the `DMA_CFG` field to handle erroneous overrun.

If adding an additional job to an existing PVP memory pipe job, the new job descriptor replaces the DUMMY descriptor (descriptor m) in memory. This updated descriptor m must now link to a new DUMMY descriptor (descriptor j).

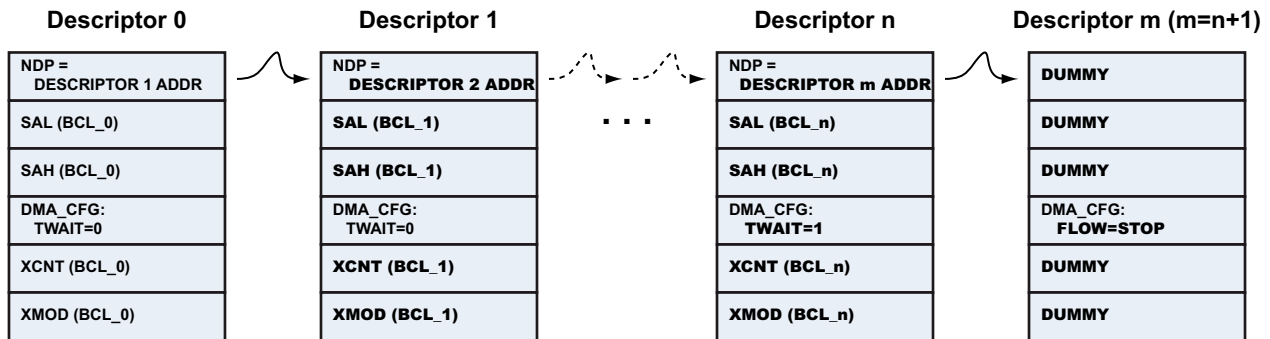


**Figure 30-54:** Linked Descriptor List for Data OUT (Results) Enabling Dynamic Extension (Back-to-Back Mode)

After completing the descriptor m and the job list extension to Data In and Data Out, clear the `TWAIT` field of the `DMA_CFG` field of descriptor n.

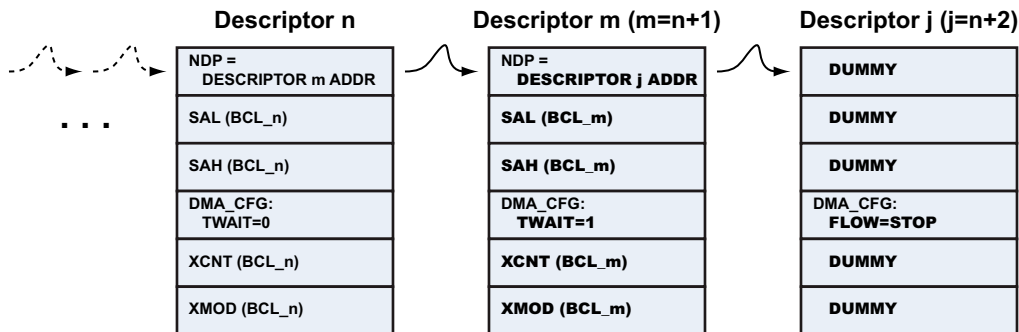


**Figure 30-55:** Dynamic Extending Linked Descriptor List for Data OUT (Job List) in Back-to-Back Mode

## Data Out Flow (Dynamic Job List, Auto-Completion Mode)

In auto-completion mode the DMA handling the Data Out (DMA 42) triggers (synchronizes) the DMA handling the Data In (DMA 43). Therefore DMA 42 has to generate a respective trigger. Configure the TRU to route the trigger of DMA 42 to the input trigger of DMA 43.

All descriptors must have `TRIG = 1` in the `DMA_CFG` field to enable trigger generation. Only the last valid descriptor has `TRIG = 0` in the `DMA_CFG` field which puts the DMA 43 into pause state.

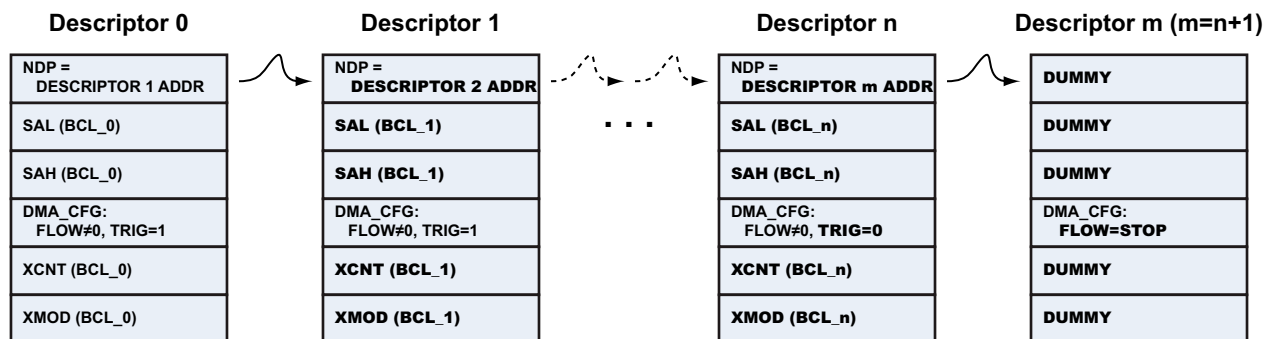


**Figure 30-56:** Linked Descriptor List for Data OUT (Results) Enabling Dynamic Extension (Auto-Completion Mode)

If adding an additional job to an existing PVP memory pipe job, this new job descriptor replaces the DUMMY descriptor (descriptor m) in memory. This updated descriptor m must now link to a new DUMMY descriptor (descriptor j).

After completing the descriptor m and the job list extension to Data In and Data Out, change the `TRIG` setting of descriptor n to `TRIG = 1`.

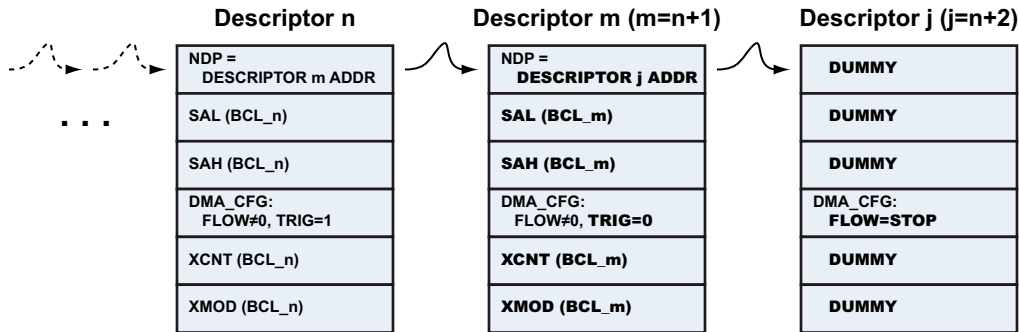


**Figure 30-57:** Dynamic Extending Linked Descriptor List for Data OUT (Job List) in Auto-Completion Mode

## Data In Descriptor List (Static Job List)

After successfully setting up and starting DMA 42, DMA 45, and (optionally) DMA 44, set up the DMA 43 linked descriptor list for the input data (Data In) to be moved into IPF1.

A third linked descriptor list that handles the Data In is set up according to the previous description of the DMA\_42 linked description list. This data stream is handled by a dedicated DMA (DMA 43) which is controlled by a linked descriptor list. The last valid descriptor of the linked list must contain a DMA\_CFG field with `FLOW = 0` (STOP Mode) to gracefully stop the DMA 43 after completing the PVP memory pipe job List and avoid loading non valid data into PVP Memory Pipe configuration register. All DMA\_CFG descriptor fields within the linked list have `FLOW = 6` or `FLOW = 7` (descriptor list mode) or alternatively `FLOW = 4` (descriptor array mode). Configure the TRU to route the trigger of DMA 45 (back-to-back mode) or DMA 42 (auto-completion mode) to input trigger of DMA 43. All DMA\_CFG descriptor field must therefore have a `TWAIT` field set (`TWAIT = 1`).

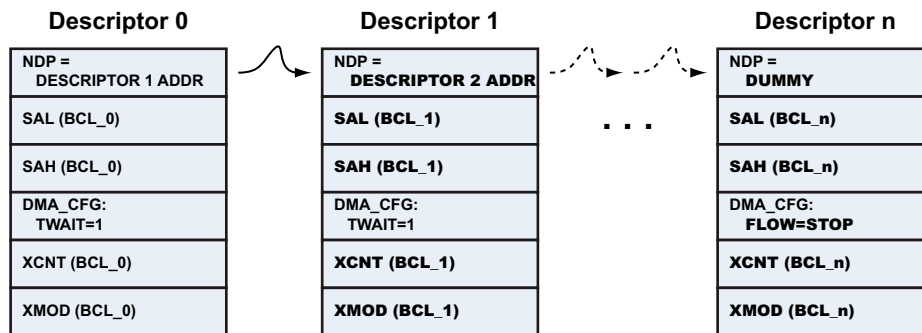


**Figure 30-58:** Static Linked Descriptor List for Data IN

## Data In Flow (Dynamic Job List)

Dynamic job lists require that the last valid descriptor (descriptor n) for DMA 43 contain a DMA\_CFG field having a `TWAIT = 1` (halt and wait for incoming trigger) instead of `FLOW = STOP`. This is so the DMA channel pauses instead of stops, and avoids loading non valid data (job creation not finalized) into the PVP memory pipe configuration register. An additional last descriptor (descriptor m) is reserved in memory

which contains dummy values and a `FLOW = 0` (STOP mode) in the `DMA_CFG` field to handle erroneous over-runs.

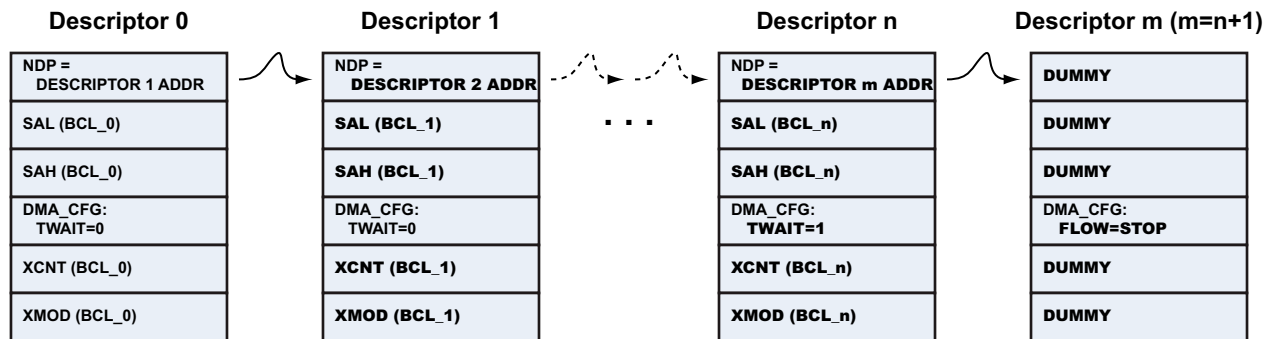


**Figure 30-59:** Linked Descriptor List for Data IN Enabling Dynamic Extension

If adding an additional job to an existing PVP memory pipe job, this new job descriptor has to replace the DUMMY descriptor (descriptor m) in memory. This updated descriptor m must now link to a new DUMMY descriptor (descriptor j).

Auto-completion mode and back-to-back mode require identical Data Out descriptor list.

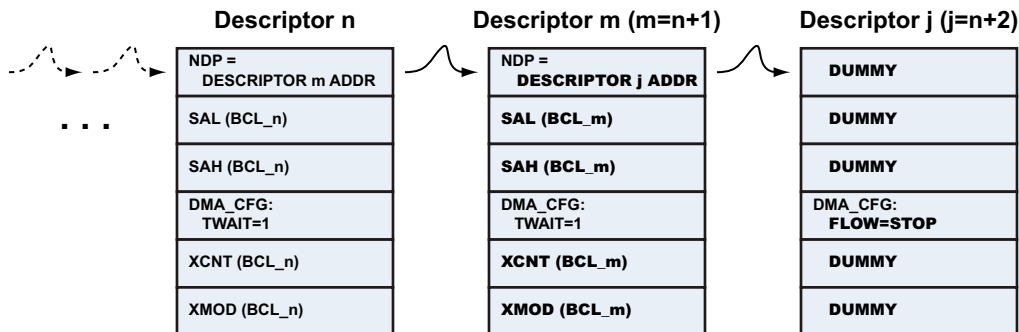


**Figure 30-60:** Dynamic Extending Linked Descriptor List for Data IN

## Starting a Job List

After setting up the three DMA linked lists, the memory pipe has to be activated (`PVP_CTL.MPEN = 1`).

## Data Flow (Static Job List, Back-to-Back Mode)

The following figure shows the data flow using a static job list and back-to-back mode.

The DMA45 generates a trigger for DMA 43 and DMA 42. DMA45 in return is triggered by the memory pipe requesting a new configuration. In this case the PVP memory pipe is enabled and the `IPF1_CFG` register bit field `FCNT = 0` (memory pipe request).



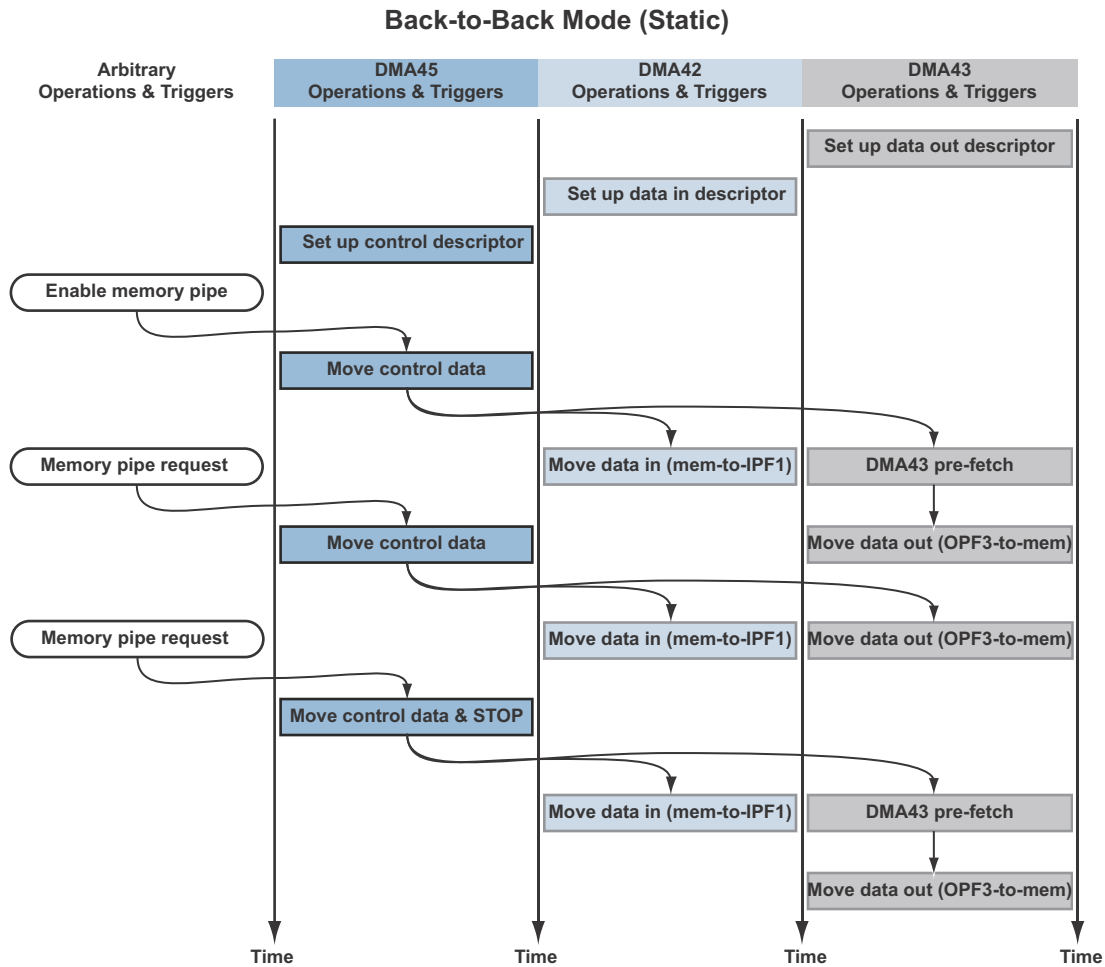


**Figure 30-61:** Static Job List Data Flow (Back-to-Back Mode)

## Data Flow (Static Job List, Auto completion Mode)

The following figure and procedure shows the data flow using a static job list and auto-completion mode.

1. DMA45 generates a trigger for DMA 43
2. DMA 43 generates a trigger for DMA 42
3. Software generates the first trigger for DMA 42
4. DMA45 is triggered when the memory pipe requests a new configuration. In this case the PVP memory pipe is enabled and the `IPF1_CFG` register bit field `FCNT = 0` (memory pipe request)

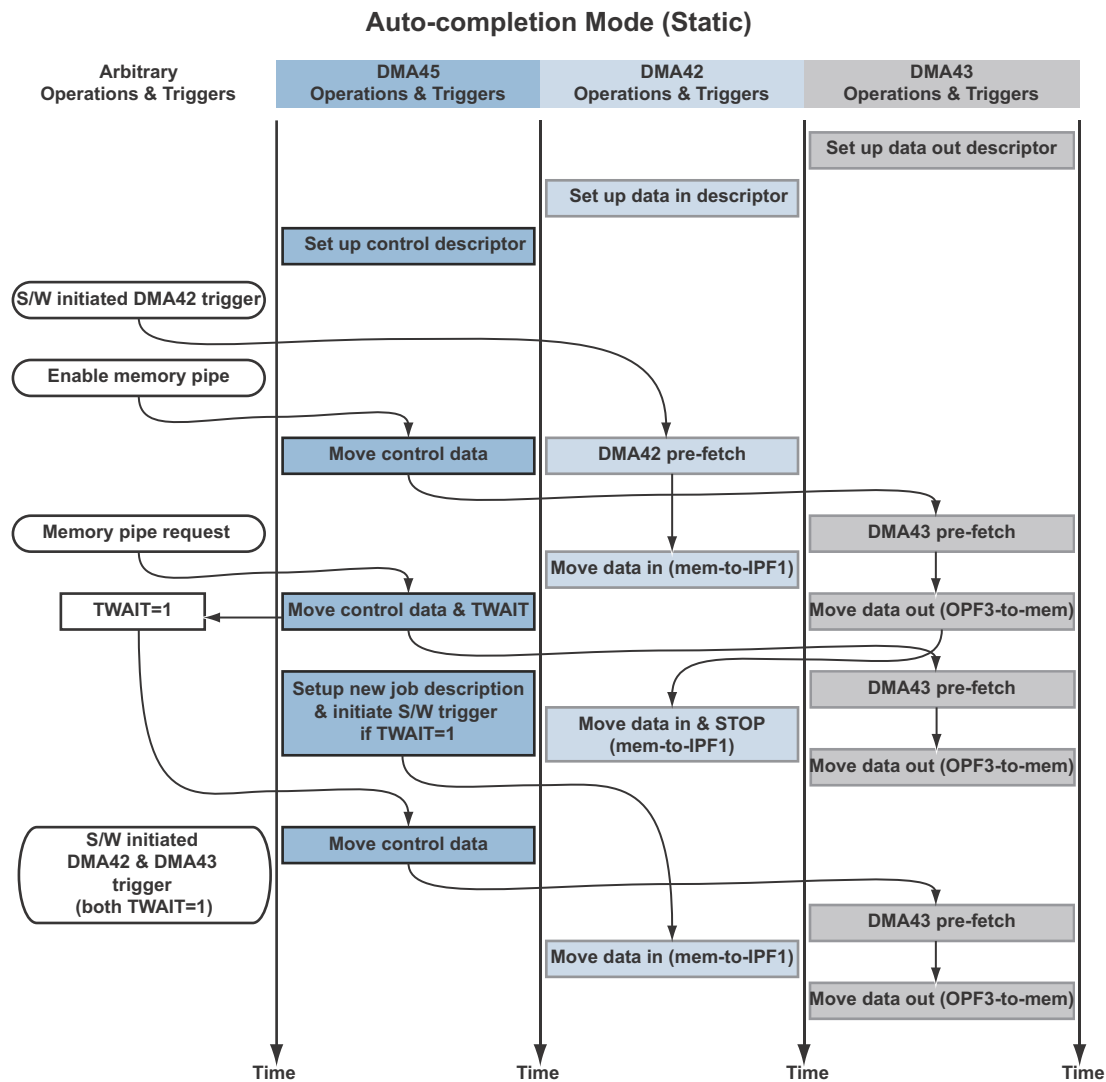


**Figure 30-62:** Static Job List Data Flow (Auto-Completion Mode)

**Data Flow (Dynamic Job List, Back-to-Back Mode)**

The following figure shows the data flow using a dynamic job list and back-to-back mode. This figure illustrates that job creation processed by the CPU is slower than the PVP. The PVP reaches the end of the valid job list before the CPU generates a new job.

The software reads  $TWAIT$  of DMA45 and as a consequence generates a trigger by software to start the new job in DMA45.

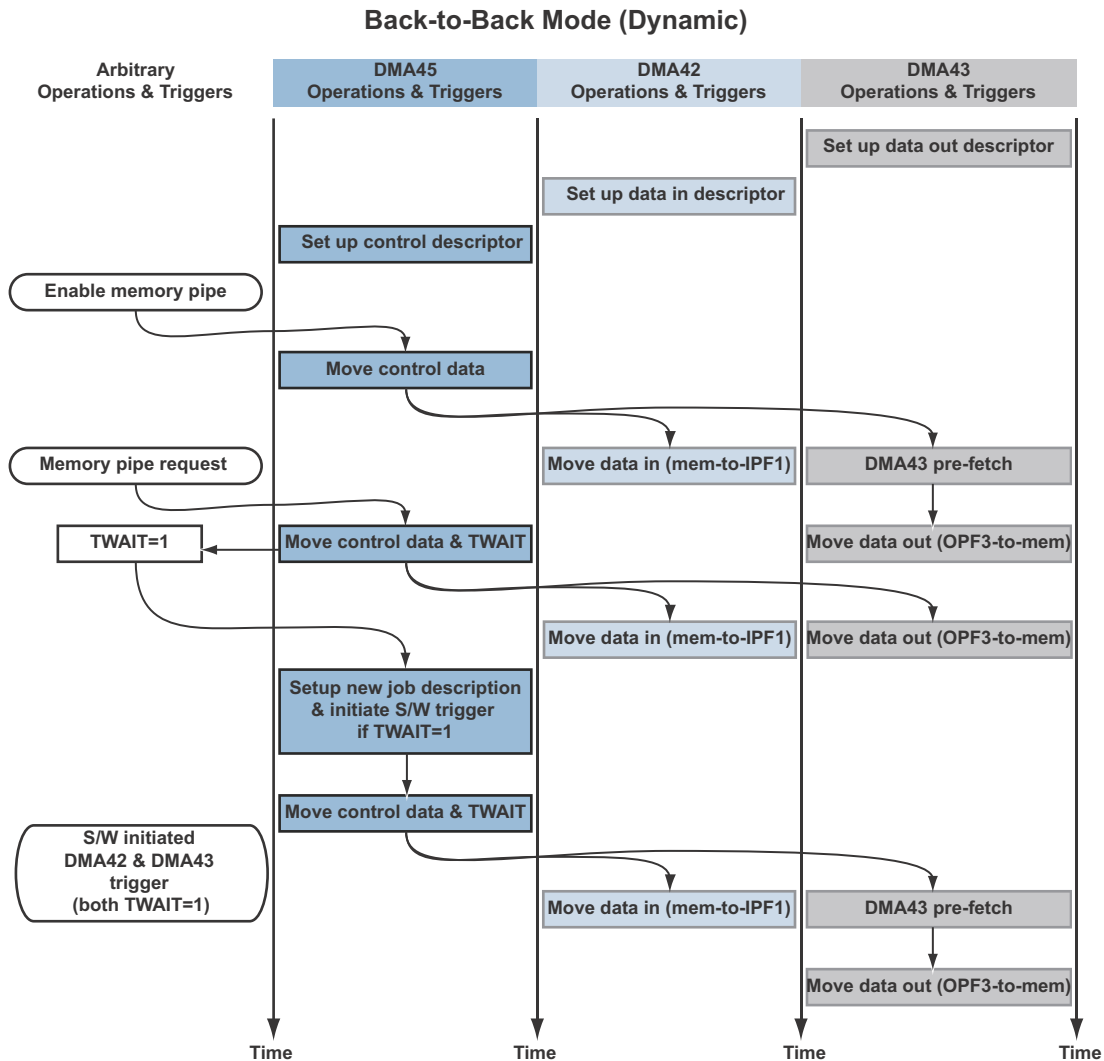


**Figure 30-63:** Dynamic Job List Data Flow (Back-to-Back Mode)

## Data Flow (Dynamic Job List, Auto-Completion)

The following figure shows the data flow using a dynamic job list and auto-completion mode. This figure and procedure demonstrate that job creation processed by the CPU is slower than the PVP processing the valid job list. Therefore, the PVP reaches the end of the valid job list before the CPU generates a new job.

1. The software reads `TWAIT` of DMA45 and as a consequence generates a trigger to start the new job in DMA45.
2. The software then reads `TWAIT` of DMA43 and as a consequence generates a trigger to start the new job in DMA42.

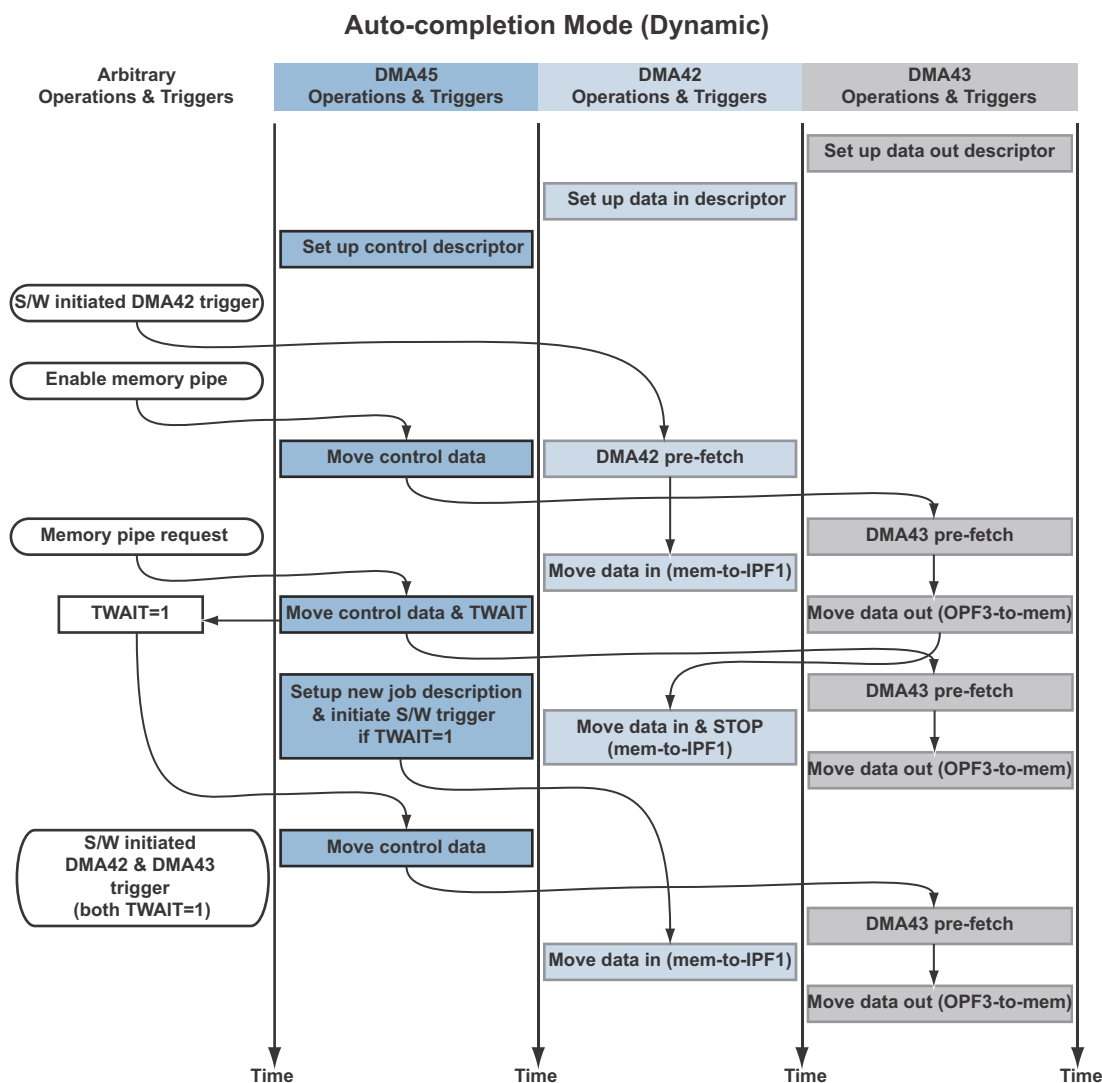


**Figure 30-64:** Dynamic Job List Data Flow (Auto-Completion Mode)

## Status (Histogram) Reports

Beside the PVP\_STAT register, which flags synchronization events and errors, the PVP also provides an automatic method to output status reports (primarily to output the histogram results). The THCx blocks latch the status of their histogram counters into MMR space every frame. Software can rely on histogram ready events as signaled by the PVP\_STAT.THCxRDY status flags and then manually collect the numerous MMR registers. To ensure consistency of the reports the MMR reads can be performed by an atomic procedure ensuring interrupts cannot interfere.

The PVP features two status output DMA channels that output the histogram reports more easily. Once configured and enabled, the status DMA operate independently from software. Report Consistency and proper read-out timing is ensured by hardware.

One status DMA is associated with the camera pipe, and one with the memory pipe. The feature is enabled by the `PVP_IPFx_PIECTL.STATEN` control bits. If enabled, a status report is sent every time `PVP_IPFx_FCNT` counts to zero. If `PVP_IPFx_FCNT` has been programmed to zero, the report is sent after every frame.

If PVP configuration is frequently updated on the fly, system software might be exposed to risk to lose track of which configuration a specific report belongs to. For this purpose the input formatters feature the `PVP_IPFx_TAG` registers. At configuration time any 16-bit tag can be loaded into these registers. If `PVP_IPFx_FCNT` expires the tag value progresses to the `PVP_IPFx_TAG_STAT` counterpart register in the MMR space. It can then optionally be stored via DMA paired with the histogram results.

## Status Word Counters

When a report is sent to status output DMA the content of the report can be structured by the program. Only the IPFx and THCx modules can contribute to the report as follows.

- IPF0 can send only to camera pipe status DMA
- IPF1 only can send to memory pipe status DMA
- THC0 and THC1 always send the status to the status DMA of the pipe they belong to

If the `PVP_IPFx_PIECTL.STATEN` status enable bit is set, the `PVP_IPFx_CFG.STATWCNT` field controls how many status words the respective input formatter sends to the status channel. Since the input formatters feature only one status word contained in the `PVP_IPFx_TAG_STAT` register, only two values are valid for the `PVP_IPFx_CFG.STATWCNT` field. If `=0`, the input formatter does not contribute to the status report. If `=1`, the input formatter sends one 32-bit word, the 32-bit padded `PVP_IPFx_TAG_STAT` register to the status channel.

**NOTE:** Since the `STATWCNT` field resides in the `PVP_IPFx_CFG` register, its value cannot be altered without setting the `DRAIN` bit `=1`. The `STATEN` bit, however, resides in the same register as the `DRAIN` bit. Therefore, the structure of the status report must not change while `DRAIN = 0`. However, `STATWCNT` is allowed to alter the `STATEN` bit with every configuration, even when `DRAIN = 0`.

The threshold blocks feature many more status registers. Each features one histogram frame counter plus 16 histogram counters (`PVP_THCx_HCNT0_STAT` to `PVP_THCx_HCNT15_STAT`) plus one `PVP_THCx_RREP_STAT` register that holds the number of run-length reports per frame.

- If the `PVP_THCx_CFG.STATWCNT` field `=0`, the respective threshold block does not contribute to the status report.
- If `STATWCNT = 1`, only the `PVP_THCx_FCNT_STAT` is sent.
- If `STATWCNT = 9`, then `PVP_THCx_HCNT0_STAT` to `PVP_THCx_HCNT7_STAT` registers are sent on top of `PVP_THCx_FCNT_STAT`.
- If `=17`, then the `PVP_THCx_FCNT_STAT` register plus all 16 histogram values are sent.
- If 18, then all 18 status registers of the Threshold block are sent.

The `STATWCNT` field of all processing blocks other than IPFs and THCs are reserved and must be initialized with zero values.

Status reports only output status registers. There is no programmable offset as in the case of block configuration.

## Block Status Structure

When a processing block contributes to a status report, its data words are headed by a block status header (BSH). The BSH is similar to the block configuration header (BCH). It consists of the block ID value and also has a word count field. The latter mirrors the `STATWCNT` value as specified in the block configuration register. Unlike BCHs the BSHs don't have an offset field. The entity of Block Status Header and respective status words is called Block Status Structure (BSS). A BSS always consists of `STATWCNT+1` 32-bit words.

A status report consists of a list of block status structures. Such a list is referred to as block status list (BSL). An example is shown in the following figure. It assumes that both threshold blocks are configured in camera pipe mode. IPF0 and THC0 are configured to send status to the report, THC1 is not and its `STATWCNT` field =0.

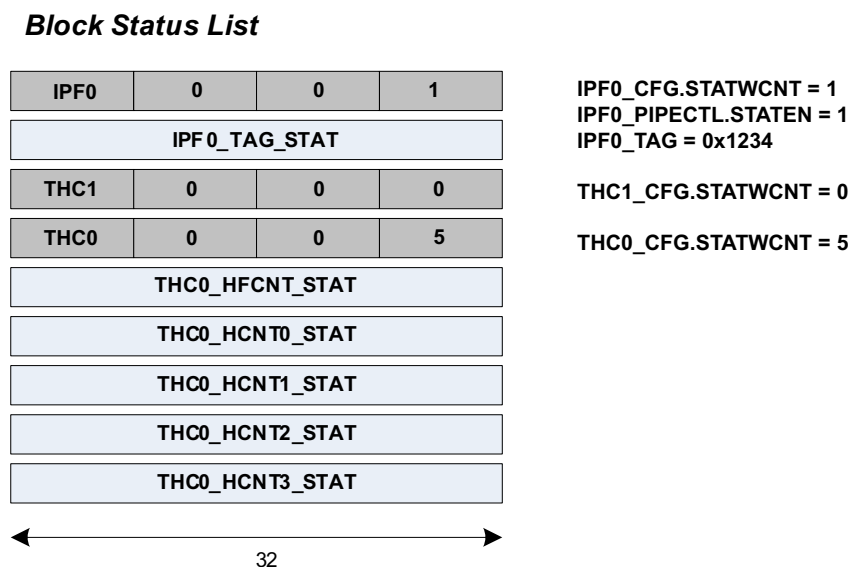


**Figure 30-65:** BSL Example

The input formatters make always the first entries in the block status list. If both threshold blocks are enabled in the same pipe, the one that comes first is closer to the IPFx in the pipe arrangement. In case the pipe configuration is not known, the block ID in the BSHs can be consulted.

Although it is not a hardware requirement, it is recommended that the work unit transfer count of the DMA associated with the status output matches the number of 32-bit words in the BSL.

## PVP Register Descriptions

PVP (PVP) contains the following registers.

**Table 30-31:** PVP Module Registers

| Name               | Description                        |
|--------------------|------------------------------------|
| PVP_REVID          | Revision ID                        |
| PVP_CTL            | Control                            |
| PVP_IMSKn          | Interrupt Mask n                   |
| PVP_STAT           | Status                             |
| PVP_ILAT           | Interrupt Latch Status n           |
| PVP_IREQn          | Interrupt Request n                |
| PVP_OPFn_CFG       | OPFn (Camera Pipe) Configuration   |
| PVP_OPFn_CTL       | OPFn (Camera Pipe) Control         |
| PVP_OPF3_CFG       | OPF3 (Memory Pipe) Configuration   |
| PVP_OPF3_CTL       | OPF3 (Memory Pipe) Control         |
| PVP_PEC_CFG        | PEC Configuration                  |
| PVP_PEC_CTL        | PEC Control                        |
| PVP_PEC_D1TH0      | PEC Lower Hysteresis Threshold     |
| PVP_PEC_D1TH1      | PEC Upper Hysteresis Threshold     |
| PVP_PEC_D2TH0      | PEC Weak Zero Crossing Threshold   |
| PVP_PEC_D2TH1      | PEC Strong Zero Crossing Threshold |
| PVP_IIMn_CFG       | IIMn Configuration                 |
| PVP_IIMn_CTL       | IIMn Control                       |
| PVP_IIMn_SCALE     | IIMn Scaling Values                |
| PVP_IIMn_SOVF_STAT | IIMn Signed Overflow Status        |

**Table 30-31:** PVP Module Registers (Continued)

| Name               | Description                                |
|--------------------|--------------------------------------------|
| PVP_IIMn_UOVF_STAT | IIMn Unsigned Overflow Status              |
| PVP_ACU_CFG        | ACU Configuration                          |
| PVP_ACU_CTL        | ACU Control                                |
| PVP_ACU_OFFSET     | ACU SUM Constant                           |
| PVP_ACU_FACTOR     | ACU PROD Constant                          |
| PVP_ACU_SHIFT      | ACU Shift Constant                         |
| PVP_ACU_MIN        | ACU Lower Sat Threshold Min                |
| PVP_ACU_MAX        | ACU Upper Sat Threshold Max                |
| PVP_UDS_CFG        | UDS Configuration                          |
| PVP_UDS_CTL        | UDS Control                                |
| PVP_UDS_OHCNT      | UDS Output HCNT                            |
| PVP_UDS_OVCNT      | UDS Output VCNT                            |
| PVP_UDS_HAVG       | UDS HAVG                                   |
| PVP_UDS_VAVG       | UDS VAVG                                   |
| PVP_IPF0_CFG       | IPF0 (Camera Pipe) Configuration           |
| PVP_IPFn_PIECTL    | IPFn (Camera/Memory Pipe) Pipe Control     |
| PVP_IPFn_CTL       | IPFn (Camera/Memory Pipe) Control          |
| PVP_IPFn_TAG       | IPFn (Camera/Memory Pipe) TAG Value        |
| PVP_IPFn_FCNT      | IPFn (Camera/Memory Pipe) Frame Count      |
| PVP_IPFn_HCNT      | IPFn (Camera/Memory Pipe) Horizontal Count |
| PVP_IPFn_VCNT      | IPFn (Camera/Memory Pipe) Vertical Count   |
| PVP_IPF0_HPOS      | IPF0 (Camera Pipe) Horizontal Position     |



**Table 30-31:** PVP Module Registers (Continued)

| Name              | Description                          |
|-------------------|--------------------------------------|
| PVP_IPF0_VPOS     | IPF0 (Camera Pipe) Vertical Position |
| PVP_IPFn_TAG_STAT | IPFn (Camera/Memory Pipe) TAG Status |
| PVP_IPF1_CFG      | IPF1 (Memory Pipe) Configuration     |
| PVP_CNVn_CFG      | CNVn Configuration                   |
| PVP_CNVn_CTL      | CNVn Control                         |
| PVP_CNVn_C00C01   | CNVn Coefficients 0,0 and 0,1        |
| PVP_CNVn_C02C03   | CNVn Coefficients 0,2 and 0,3        |
| PVP_CNVn_C04      | CNVn Coefficient 0,4                 |
| PVP_CNVn_C10C11   | CNVn Coefficients 1,0 and 1,1        |
| PVP_CNVn_C12C13   | CNVn Coefficients 1,2 and 1,3        |
| PVP_CNVn_C14      | CNVn Coefficient 1,4                 |
| PVP_CNVn_C20C21   | CNVn Coefficients 2,0 and 2,1        |
| PVP_CNVn_C22C23   | CNVn Coefficients 2,2 and 2,3        |
| PVP_CNVn_C24      | CNVn Coefficient 2,4                 |
| PVP_CNVn_C30C31   | CNVn Coefficients 3,0 and 3,1        |
| PVP_CNVn_C32C33   | CNVn Coefficients 3,2 and 3,3        |
| PVP_CNVn_C34      | CNVn Coefficient 3,4                 |
| PVP_CNVn_C40C41   | CNVn Coefficients 4,0 and 4,1        |
| PVP_CNVn_C42C43   | CNVn Coefficients 4,2 and 4,3        |
| PVP_CNVn_C44      | CNVn Coefficient 4,4                 |
| PVP_CNVn_SCALE    | CNVn Scaling Factor                  |
| PVP_THCn_CFG      | THCn Configuration                   |

**Table 30-31:** PVP Module Registers (Continued)

| Name             | Description                |
|------------------|----------------------------|
| PVP_THCn_CTL     | THCn Control               |
| PVP_THCn_HFCNT   | THCn Histogram Frame Count |
| PVP_THCn_RMAXREP | THCn Max RLE Reports       |
| PVP_THCn_CMINVAL | THCn Min Clip Value        |
| PVP_THCn_CMINTH  | THCn Clip Min Threshold    |
| PVP_THCn_CMAXTH  | THCn Clip Max Threshold    |
| PVP_THCn_CMAXVAL | THCn Max Clip Value        |
| PVP_THCn_TH0     | THCn Threshold Value 0     |
| PVP_THCn_TH1     | THCn Threshold Value 1     |
| PVP_THCn_TH2     | THCn Threshold Value 2     |
| PVP_THCn_TH3     | THCn Threshold Value 3     |
| PVP_THCn_TH4     | THCn Threshold Value 4     |
| PVP_THCn_TH5     | THCn Threshold Value 5     |
| PVP_THCn_TH6     | THCn Threshold Value 6     |
| PVP_THCn_TH7     | THCn Threshold Value 7     |
| PVP_THCn_TH8     | THCn Threshold Value 8     |
| PVP_THCn_TH9     | THCn Threshold Value 9     |
| PVP_THCn_TH10    | THCn Threshold Value 10    |
| PVP_THCn_TH11    | THCn Threshold Value 11    |
| PVP_THCn_TH12    | THCn Threshold Value 12    |
| PVP_THCn_TH13    | THCn Threshold Value 13    |
| PVP_THCn_TH14    | THCn Threshold Value 14    |

**Table 30-31:** PVP Module Registers (Continued)

| Name                 | Description                        |
|----------------------|------------------------------------|
| PVP_THCn_TH15        | THCn Threshold Value 15            |
| PVP_THCn_HHPOS       | THCn Histogram Horizontal Position |
| PVP_THCn_HVPOS       | THCn Histogram Vertical Position   |
| PVP_THCn_HHCNT       | THCn Histogram Horizontal Count    |
| PVP_THCn_HVCNT       | THCn Histogram Vertical Count      |
| PVP_THCn_RHPOS       | THCn RLE Horizontal Position       |
| PVP_THCn_RVPOS       | THCn RLE Vertical Position         |
| PVP_THCn_RHCNT       | THCn RLE Horizontal Count          |
| PVP_THCn_RVCNT       | THCn RLE Vertical Count            |
| PVP_THCn_HFCNT_STAT  | THCn Histogram Frame Count Status  |
| PVP_THCn_HCNT0_STAT  | THCn Histogram Counter Value 0     |
| PVP_THCn_HCNT1_STAT  | THCn Histogram Counter Value 1     |
| PVP_THCn_HCNT2_STAT  | THCn Histogram Counter Value 2     |
| PVP_THCn_HCNT3_STAT  | THCn Histogram Counter Value 3     |
| PVP_THCn_HCNT4_STAT  | THCn Histogram Counter Value 4     |
| PVP_THCn_HCNT5_STAT  | THCn Histogram Counter Value 5     |
| PVP_THCn_HCNT6_STAT  | THCn Histogram Counter Value 6     |
| PVP_THCn_HCNT7_STAT  | THCn Histogram Counter Value 7     |
| PVP_THCn_HCNT8_STAT  | THCn Histogram Counter Value 8     |
| PVP_THCn_HCNT9_STAT  | THCn Histogram Counter Value 9     |
| PVP_THCn_HCNT10_STAT | THCn Histogram Counter Value 10    |
| PVP_THCn_HCNT11_STAT | THCn Histogram Counter Value 11    |

Table 30-31: PVP Module Registers (Continued)

| Name                 | Description                     |
|----------------------|---------------------------------|
| PVP_THCn_HCNT12_STAT | THCn Histogram Counter Value 12 |
| PVP_THCn_HCNT13_STAT | THCn Histogram Counter Value 13 |
| PVP_THCn_HCNT14_STAT | THCn Histogram Counter Value 14 |
| PVP_THCn_HCNT15_STAT | THCn Histogram Counter Value 15 |
| PVP_THCn_RREP_STAT   | THCn Number of RLE Reports      |
| PVP_PMA_CFG          | PMA Configuration               |

Revision ID

The PVP\_REVID register provides the PVP module revision information.

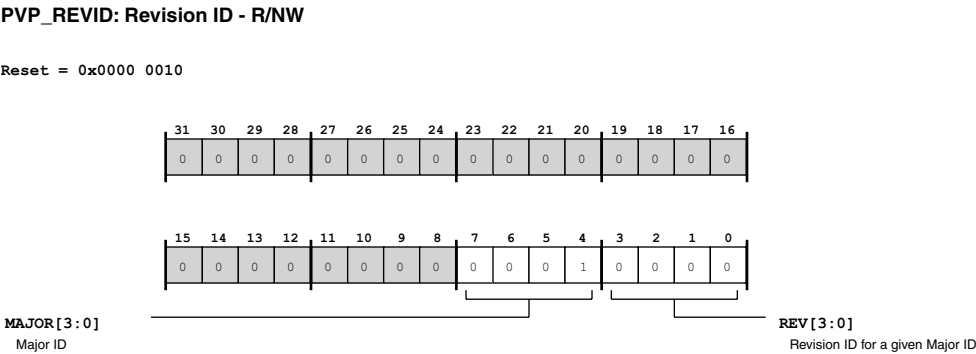


Figure 30-66: PVP\_REVID Register Diagram

Table 30-32: PVP\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration           |
|---------|----------|-----------------------------------|
| 7:4     | MAJOR    | Major ID.                         |
| 3:0     | REV      | Revision ID for a given Major ID. |

Control

The PVP\_CTL register provides enables the module, enables the memory pipe, enables the camera pipe, and selects the SCLK0-to-PVPCLK clock divisor. Note that PVP\_CTL.PVPEN, PVP\_CTL.MPEN, and/or

PVP\_CTL.CPEN disable and re-enable MUST be followed by a fresh write of all required values in all shadow registers.

**PVP\_CTL: Control - R/W**

Reset = 0x0000 0000

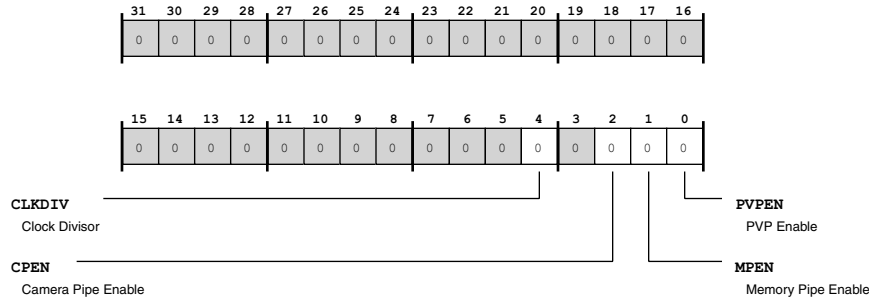


**Figure 30-67:** PVP\_CTL Register Diagram

**Table 30-33:** PVP\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | CLKDIV   | Clock Divisor.<br>The PVP_CTL.CLKDIV selects the SCLK0-to-PVPCLK clock divisor as:<br>$\text{PVPCLK} = \text{SCLK0} / (2^{\text{PVP\_CTL.CLKDIV}})$ |
|         |          | 0 PVPCLK = SCLK0                                                                                                                                    |
|         |          | 1 PVPCLK = SCLK0/2                                                                                                                                  |
| 2       | CPEN     | Camera Pipe Enable.<br>The PVP_CTL.CPEN enables or disables all processing blocks in the camera pipe.                                               |
|         |          | 0 Disable Camera Pipe                                                                                                                               |
|         |          | 1 Enable Camera Pipe                                                                                                                                |
| 1       | MPEN     | Memory Pipe Enable.<br>The PVP_CTL.MPEN enables/disables all processing blocks in the memory pipe.                                                  |
|         |          | 0 Disable Memory Pipe                                                                                                                               |
|         |          | 1 Enable Memory Pipe                                                                                                                                |

**Table 30-33:** PVP\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                    |             |
|---------|----------|------------------------------------------------------------|-------------|
| 0       | PVPEN    | PVP Enable.<br>The PVP_CTL.PVPEN enables/disables the PVP. |             |
|         |          | 0                                                          | Disable PVP |
|         |          | 1                                                          | Enable PVP  |

**Interrupt Mask n**

The PVP\_IMSKn register enables (unmasks) or disables (masks) status related interrupts for the PVP. For all PVP\_IMSKn bits, setting the bit enables the interrupt, and clearing the bit disables the interrupt.

**PVP\_IMSKn: Interrupt Mask n - R/W**

Reset = 0x0000 0000

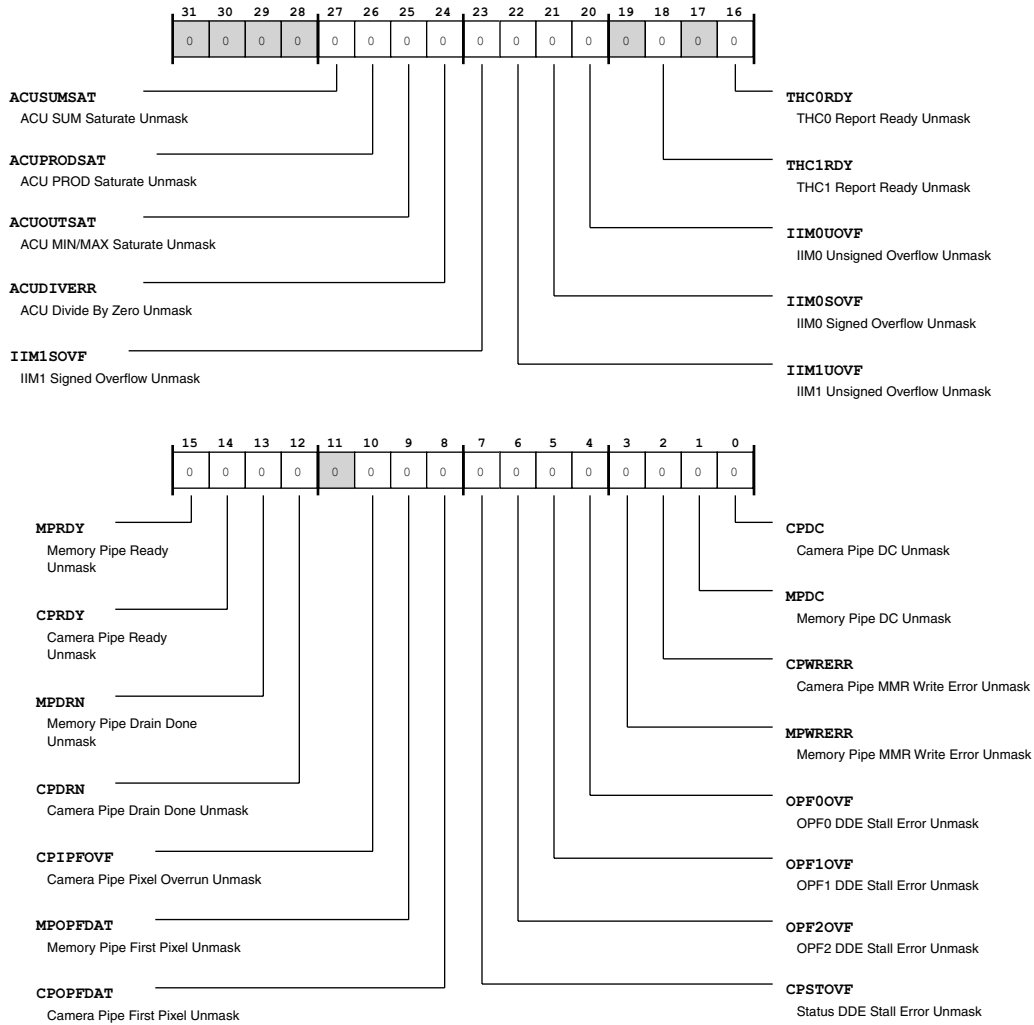


**Figure 30-68:** PVP\_IMSKn Register Diagram

**Table 30-34:** PVP\_IMSKn Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                        |
|---------|------------|----------------------------------------------------------------------------------------------------------------|
| 27      | ACUSUMSAT  | ACU SUM Saturate Unmask.<br>The PVP_IMSKn.ACUSUMSAT bit enables/disables the PVP_IMSKn.ACUSUMSAT interrupt.    |
| 26      | ACUPRODSAT | ACU PROD Saturate Unmask.<br>The PVP_IMSKn.ACUPRODSAT bit enables/disables the PVP_IMSKn.ACUPRODSAT interrupt. |

**Table 30-34:** PVP\_IMSKn Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                         |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------|
| 25      | ACUOUTSAT | ACU MIN/MAX Saturate Unmask.<br>The PVP_IMSKn.ACUOUTSAT bit enables/disables the PVP_IMSKn.ACUOUTSAT interrupt. |
| 24      | ACUDIVERR | ACU Divide By Zero Unmask.<br>The PVP_IMSKn.ACUDIVERR bit enables/disables the PVP_IMSKn.ACUDIVERR interrupt.   |
| 23      | IIM1SOVF  | IIM1 Signed Overflow Unmask.<br>The PVP_IMSKn.IIM1SOVF bit enables/disables the PVP_IMSKn.IIM1SOVF interrupt.   |
| 22      | IIM1UOVF  | IIM1 Unsigned Overflow Unmask.<br>The PVP_IMSKn.IIM1UOVF bit enables/disables the PVP_IMSKn.IIM1UOVF interrupt. |
| 21      | IIM0SOVF  | IIM0 Signed Overflow Unmask.<br>The PVP_IMSKn.IIM0SOVF bit enables/disables the PVP_IMSKn.IIM0SOVF interrupt.   |
| 20      | IIM0UOVF  | IIM0 Unsigned Overflow Unmask.<br>The PVP_IMSKn.IIM0UOVF bit enables/disables the PVP_IMSKn.IIM0UOVF interrupt. |
| 18      | THC1RDY   | THC1 Report Ready Unmask.<br>The PVP_IMSKn.THC1RDY bit enables/disables the PVP_IMSKn.THC1RDY interrupt.        |
| 16      | THC0RDY   | THC0 Report Ready Unmask.<br>The PVP_IMSKn.THCORDY bit enables/disables the PVP_IMSKn.THCORDY interrupt.        |
| 15      | MPRDY     | Memory Pipe Ready Unmask.<br>The PVP_IMSKn.MPRDY bit enables/disables the PVP_IMSKn.MPRDY interrupt.            |
| 14      | CPRDY     | Camera Pipe Ready Unmask.<br>The PVP_IMSKn.CPRDY bit enables/disables the PVP_IMSKn.CPRDY interrupt.            |
| 13      | MPDRN     | Memory Pipe Drain Done Unmask.<br>The PVP_IMSKn.MPDRN bit enables/disables the PVP_IMSKn.MPDRN interrupt.       |



**Table 30-34:** PVP\_IMSKn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------|
| 12      | CPDRN    | Camera Pipe Drain Done Unmask.<br>The PVP_IMSKn.CPDRN bit enables/disables the PVP_IMSKn.CPDRN interrupt.          |
| 10      | CPIPF0VF | Camera Pipe Pixel Overrun Unmask.<br>The PVP_IMSKn.CPIPF0VF bit enables/disables the PVP_IMSKn.CPIPF0VF interrupt. |
| 9       | MPOPFDAT | Memory Pipe First Pixel Unmask.<br>The PVP_IMSKn.MPOPFDAT bit enables/disables the PVP_IMSKn.MPOPFDAT interrupt.   |
| 8       | CPOPFDAT | Camera Pipe First Pixel Unmask.<br>The PVP_IMSKn.CPOPFDAT bit enables/disables the PVP_IMSKn.CPOPFDAT interrupt.   |
| 7       | CPSTOVF  | Status DDE Stall Error Unmask.<br>The PVP_IMSKn.CPSTOVF bit enables/disables the PVP_IMSKn.CPSTOVF interrupt.      |
| 6       | OPF2OVF  | OPF2 DDE Stall Error Unmask.<br>The PVP_IMSKn.OPF2OVF bit enables/disables the PVP_IMSKn.OPF2OVF interrupt.        |
| 5       | OPF1OVF  | OPF1 DDE Stall Error Unmask.<br>The PVP_IMSKn.OPF1OVF bit enables/disables the PVP_IMSKn.OPF1OVF interrupt.        |
| 4       | OPF0OVF  | OPF0 DDE Stall Error Unmask.<br>The PVP_IMSKn.OPF0OVF bit enables/disables the PVP_IMSKn.OPF0OVF interrupt.        |
| 3       | MPWRERR  | Memory Pipe MMR Write Error Unmask.<br>The PVP_IMSKn.MPWRERR bit enables/disables the PVP_IMSKn.MPWRERR interrupt. |
| 2       | CPWRERR  | Camera Pipe MMR Write Error Unmask.<br>The PVP_IMSKn.CPWRERR bit enables/disables the PVP_IMSKn.CPWRERR interrupt. |
| 1       | MPDC     | Memory Pipe DC Unmask.<br>The PVP_IMSKn.MPDC bit enables/disables the PVP_IMSKn.MPDC interrupt.                    |

**Table 30-34:** PVP\_IMSKn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 0       | CPDC     | Camera Pipe DC Unmask.<br>The PVP_IMSKn.CPDC bit enables/disables the PVP_IMSKn.CPDC interrupt. |

## Status

The PVP\_STAT register indicates whether or not a PVP status related interrupts is pending. For all PVP\_STAT bits, a set bit indicates a pending interrupt, and a cleared bit indicates that no interrupt is pending. The bits in this register are automatically set or cleared by changes to PVP status, and all bits are read-only.

# PVP\_STAT: Status - R/NW

Reset = 0x0000 c000

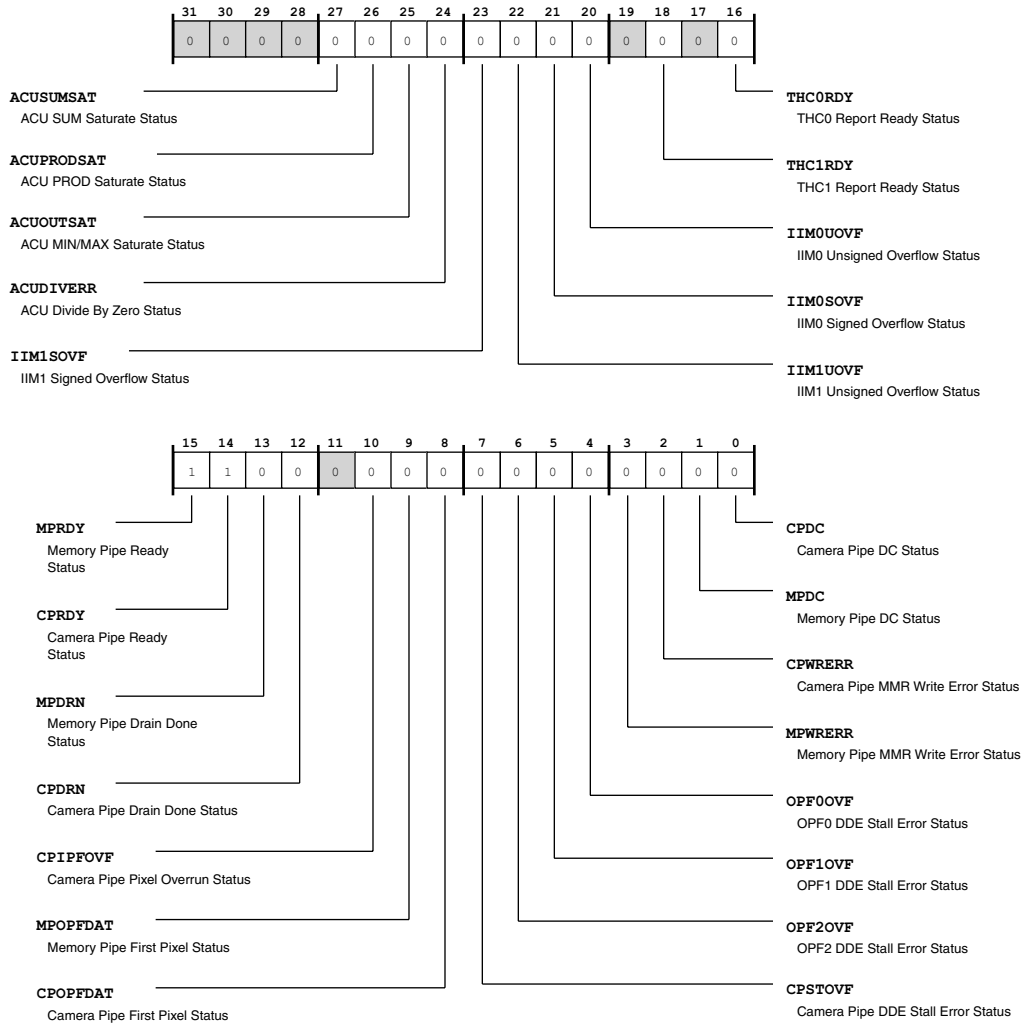


Figure 30-69: PVP\_STAT Register Diagram

**Table 30-35:** PVP\_STAT Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                      |                      |
|---------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| 27      | ACUSUMSAT  | ACU SUM Saturate Status.<br>The PVP_STAT.ACUSUMSAT bit indicates whether or not the PVP_STAT.ACUSUMSAT interrupt is pending. This interrupt is set by an ACU on a frame boundary having sum saturation, and this interrupt is cleared on the next frame boundary.            |                      |
|         |            | 0                                                                                                                                                                                                                                                                            | No Pending Interrupt |
|         |            | 1                                                                                                                                                                                                                                                                            | Pending Interrupt    |
| 26      | ACUPRODSAT | ACU PROD Saturate Status.<br>The PVP_STAT.ACUPRODSAT bit indicates whether or not the PVP_STAT.ACUPRODSAT interrupt is pending. This interrupt is set by an ACU on a frame boundary having product saturation, and this interrupt is cleared on the next frame boundary.     |                      |
|         |            | 0                                                                                                                                                                                                                                                                            | No Pending Interrupt |
|         |            | 1                                                                                                                                                                                                                                                                            | Pending Interrupt    |
| 25      | ACUOUTSAT  | ACU MIN/MAX Saturate Status.<br>The PVP_STAT.ACUOUTSAT bit indicates whether or not the PVP_STAT.ACUOUTSAT interrupt is pending. This interrupt is set by an ACU on a frame boundary having min or max saturation, and this interrupt is cleared on the next frame boundary. |                      |
|         |            | 0                                                                                                                                                                                                                                                                            | No Pending Interrupt |
|         |            | 1                                                                                                                                                                                                                                                                            | Pending Interrupt    |
| 24      | ACUDIVERR  | ACU Divide By Zero Status.<br>The PVP_STAT.ACUDIVERR bit indicates whether or not the PVP_STAT.ACUDIVERR interrupt is pending. This interrupt is set by an ACU on a frame boundary having div by zero error, and this interrupt is cleared on the next frame boundary.       |                      |
|         |            | 0                                                                                                                                                                                                                                                                            | No Pending Interrupt |
|         |            | 1                                                                                                                                                                                                                                                                            | Pending Interrupt    |

**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23      | IIM1SOVF | IIM1 Signed Overflow Status.<br>The PVP_STAT.IIM1SOVF bit indicates whether or not the PVP_STAT.IIM1SOVF interrupt is pending. This interrupt is set by an IIM every frame boundary when signed saturation occurs, and this interrupt is cleared on the next frame boundary.     |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                              |
| 22      | IIM1UOVF | IIM1 Unsigned Overflow Status.<br>The PVP_STAT.IIM1UOVF bit indicates whether or not the PVP_STAT.IIM1UOVF interrupt is pending. This interrupt is set by an IIM every frame boundary when unsigned saturation occurs, and this interrupt is cleared on the next frame boundary. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                              |
| 21      | IIM0SOVF | IIM0 Signed Overflow Status.<br>The PVP_STAT.IIM0SOVF bit indicates whether or not the PVP_STAT.IIM0SOVF interrupt is pending. This interrupt is set by an IIM every frame boundary when signed saturation occurs, and this interrupt is cleared on the next frame boundary.     |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                              |
| 20      | IIM0UOVF | IIM0 Unsigned Overflow Status.<br>The PVP_STAT.IIM0UOVF bit indicates whether or not the PVP_STAT.IIM0UOVF interrupt is pending. This interrupt is set by an IIM every frame boundary when unsigned saturation occurs, and this interrupt is cleared on the next frame boundary. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                              |

**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | THC1RDY  | THC1 Report Ready Status.<br>The PVP_STAT.THClRDY bit indicates whether or not the PVP_STAT.THClRDY interrupt is pending. This interrupt is set by THC1 on every HFCNT expiry boundary or when auto-disabled, and this interrupt is cleared at the beginning of the next line of the next frame. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                              |
| 16      | THC0RDY  | THC0 Report Ready Status.<br>The PVP_STAT.THClRDY bit indicates whether or not the PVP_STAT.THClRDY interrupt is pending. This interrupt is set by THC0 on every HFCNT expiry boundary or when auto-disabled, and this interrupt is cleared at the beginning of the next line of the next frame. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                              |
| 15      | MPRDY    | Memory Pipe Ready Status.<br>The PVP_STAT.MPRDY bit indicates whether or not the PVP_STAT.MPRDY interrupt is pending. This interrupt is set when the IPF1 application start bit is cleared, and this interrupt is cleared when the IPF1 application start bit is set.                            |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                              |
| 14      | CPRDY    | Camera Pipe Ready Status.<br>The PVP_STAT.CPRDY bit indicates whether or not the PVP_STAT.CPRDY interrupt is pending. This interrupt is set when the IPF0 application start bit is cleared, and this interrupt is cleared when the IPF0 application start bit is set.                            |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                              |

**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13      | MPDRN    | Memory Pipe Drain Done Status.<br>The <code>PVP_STAT.MPDRN</code> bit indicates whether or not the <code>PVP_STAT.MPDRN</code> interrupt is pending. This interrupt is set when the memory pipes are drained and all memory pipe modules are disabled, and this interrupt is cleared when the next memory pipe daisy chain load is started.                     |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                                             |
| 12      | CPDRN    | Camera Pipe Drain Done Status.<br>The <code>PVP_STAT.CPDRN</code> bit indicates whether or not the <code>PVP_STAT.CPDRN</code> interrupt is pending. This interrupt is set when all three camera pipes are drained and all camera pipe modules are disabled, and this interrupt is cleared when the next camera pipe daisy chain load is started.               |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                                             |
| 10      | CPIPF0VF | Camera Pipe Pixel Overrun Status.<br>The <code>PVP_STAT.CPIPF0VF</code> bit indicates whether or not the <code>PVP_STAT.CPIPF0VF</code> interrupt is pending. This interrupt is set when pixel overrun occurs in IPF0, and this interrupt is cleared when the first pixel of the next line of that frame has entered memory pipe OPF.                           |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                                             |
| 9       | MPOPFDAT | Memory Pipe First Pixel Status.<br>The <code>PVP_STAT.MPOPFDAT</code> bit indicates whether or not the <code>PVP_STAT.MPOPFDAT</code> interrupt is pending. This interrupt is set when the first pixel of the frame has entered memory pipe OPF, and this interrupt is cleared when the first pixel of the next line of that frame has entered memory pipe OPF. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                                             |

**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | CPOPFDAT | Camera Pipe First Pixel Status.<br>The PVP_STAT.CPOPFDAT bit indicates whether or not the PVP_STAT.CPOPFDAT interrupt is pending. This interrupt is set when the first pixel of the frame has entered all 3 camera pipe OPFs, and this interrupt is cleared when the first pixel of the next line of that frame has entered all 3 OPFs. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                     |
| 7       | CPSTOVF  | Camera Pipe DDE Stall Error Status.<br>The PVP_STAT.CPSTOVF bit indicates whether or not the PVP_STAT.CPSTOVF interrupt is pending. This interrupt is set when FIFO overrun occurs in status DDE, and this interrupt is cleared when first pixel of the next frame enters the FIFO.                                                     |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                     |
| 6       | OPF2OVF  | OPF2 DDE Stall Error Status.<br>The PVP_STAT.OPF2OVF bit indicates whether or not the PVP_STAT.OPF2OVF interrupt is pending. This interrupt is set when FIFO overrun occurs in OPF2, and this interrupt is cleared when first pixel of the next frame enters the FIFO.                                                                  |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                     |
| 5       | OPF1OVF  | OPF1 DDE Stall Error Status.<br>The PVP_STAT.OPF1OVF bit indicates whether or not the PVP_STAT.OPF1OVF interrupt is pending. This interrupt is set when FIFO overrun occurs in OPF1, and this interrupt is cleared when first pixel of the next frame enters the FIFO.                                                                  |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                                     |



**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | OPF0OVF  | OPF0 DDE Stall Error Status.<br>The <code>PVP_STAT.OPF0OVF</code> bit indicates whether or not the <code>PVP_STAT.OPF0OVF</code> interrupt is pending. This interrupt is set when FIFO overrun occurs in OPF0, and this interrupt is cleared when first pixel of the next frame enters the FIFO.                                 |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                              |
| 3       | MPWRERR  | Memory Pipe MMR Write Error Status.<br>The <code>PVP_STAT.MPWRERR</code> bit indicates whether or not the <code>PVP_STAT.MPWRERR</code> interrupt is pending. This interrupt is set when a MMR write was attempted to a memory pipe module during daisy chain load, and this interrupt is cleared when daisy chain load is done. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                              |
| 2       | CPWRERR  | Camera Pipe MMR Write Error Status.<br>The <code>PVP_STAT.CPWRERR</code> bit indicates whether or not the <code>PVP_STAT.CPWRERR</code> interrupt is pending. This interrupt is set when a MMR write was attempted to a camera pipe module during daisy chain load, and this interrupt is cleared when daisy chain load is done. |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                              |
| 1       | MPDC     | Memory Pipe DC Status.<br>The <code>PVP_STAT.MPDC</code> bit indicates whether or not the <code>PVP_STAT.MPDC</code> interrupt is pending. This interrupt is set when memory pipe daisy chain load status equals done, and this interrupt is cleared by daisy chain load start.                                                  |
|         |          | 0 No Pending Interrupt                                                                                                                                                                                                                                                                                                           |
|         |          | 1 Pending Interrupt                                                                                                                                                                                                                                                                                                              |

**Table 30-35:** PVP\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                            |                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------|
| 0       | CPDC     | Camera Pipe DC Status.<br>The PVP_STAT.CPDC bit indicates whether or not the PVP_STAT.CPDC interrupt is pending. This interrupt is set when all camera pipes daisy chain load equal done, and this interrupt is cleared by daisy chain load start. |                      |
|         |          | 0                                                                                                                                                                                                                                                  | No Pending Interrupt |
|         |          | 1                                                                                                                                                                                                                                                  | Pending Interrupt    |

## Interrupt Latch Status n

The PVP\_ILAT register indicates that latched status of interrupts for the PVP. For all PVP\_ILAT bits, a set the bit indicates that the interrupt has occurred, and cleared bit indicates that the interrupt has not occurred. The bits in PVP\_ILAT are set by PVP status, but are cleared with a write-1-to-clear operation. Note that clearing a bit in PVP\_ILAT clears the corresponding bit in the PVP\_IREQn register.

**PVP\_ILAT: Interrupt Latch Status n - R/W**

Reset = 0x0000 c000

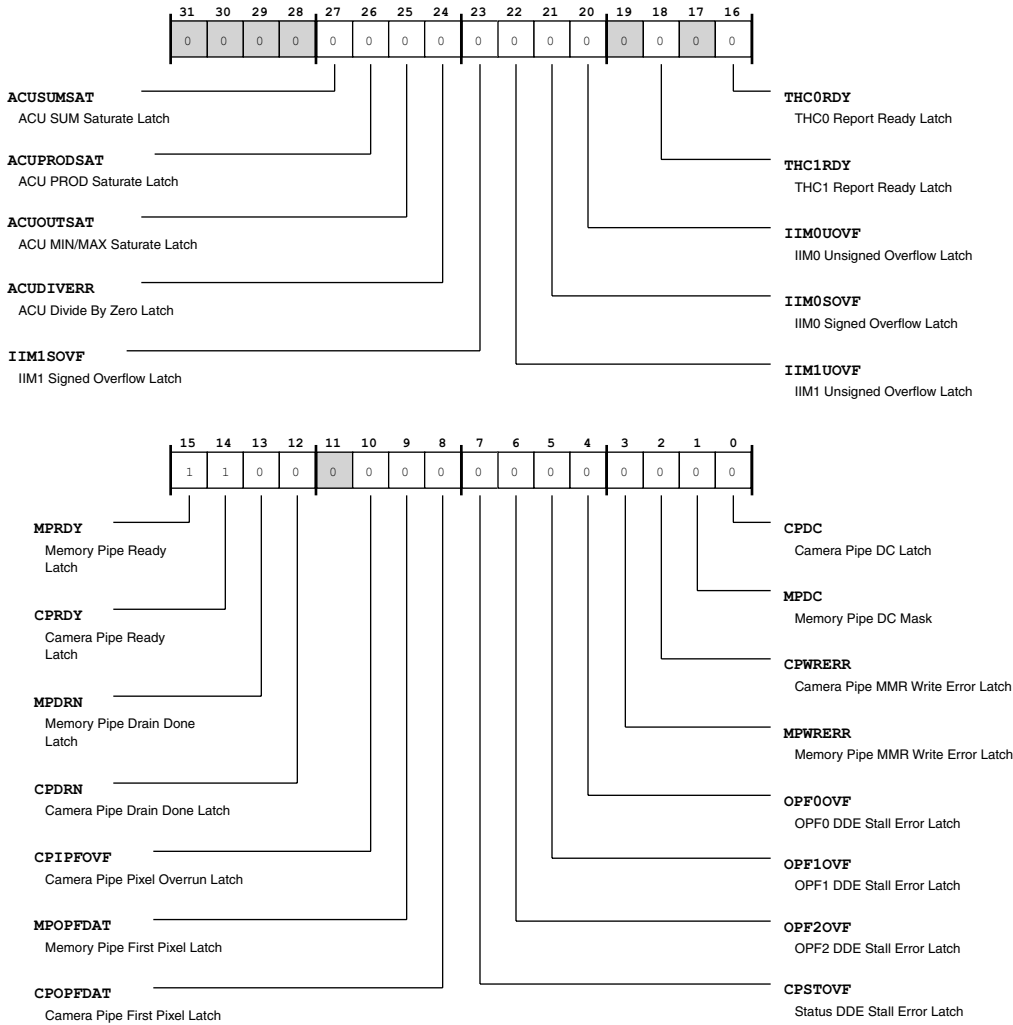


**Figure 30-70: PVP\_ILAT Register Diagram**

**Table 30-36: PVP\_ILAT Register Fields**

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                     |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------|
| 27      | ACUSUMSAT | ACU SUM Saturate Latch.<br>The PVP_ILAT.ACUSUMSAT bit indicates that the PVP_ILAT.ACUSUMSAT has occurred. Write 1 to clear this status bit. |
|         |           | 1 Interrupt Latched                                                                                                                         |

**Table 30-36:** PVP\_ILAT Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                         |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 26      | ACUPRODSAT | ACU PROD Saturate Latch.<br>The PVP_ILAT.ACUPRODSAT bit indicates that the PVP_ILAT.ACUPRODSAT has occurred. Write 1 to clear this status bit.  |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 25      | ACUOUTSAT  | ACU MIN/MAX Saturate Latch.<br>The PVP_ILAT.ACUOUTSAT bit indicates that the PVP_ILAT.ACUOUTSAT has occurred. Write 1 to clear this status bit. |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 24      | ACUDIVERR  | ACU Divide By Zero Latch.<br>The PVP_ILAT.ACUDIVERR bit indicates that the PVP_ILAT.ACUDIVERR has occurred. Write 1 to clear this status bit.   |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 23      | IIM1SOVF   | IIM1 Signed Overflow Latch.<br>The PVP_ILAT.IIM1SOVF bit indicates that the PVP_ILAT.IIM1SOVF has occurred. Write 1 to clear this status bit.   |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 22      | IIM1UOVF   | IIM1 Unsigned Overflow Latch.<br>The PVP_ILAT.IIM1UOVF bit indicates that the PVP_ILAT.IIM1UOVF has occurred. Write 1 to clear this status bit. |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 21      | IIM0SOVF   | IIM0 Signed Overflow Latch.<br>The PVP_ILAT.IIM0SOVF bit indicates that the PVP_ILAT.IIM0SOVF has occurred. Write 1 to clear this status bit.   |
|         |            | 1 Interrupt Latched                                                                                                                             |
| 20      | IIM0UOVF   | IIM0 Unsigned Overflow Latch.<br>The PVP_ILAT.IIM0UOVF bit indicates that the PVP_ILAT.IIM0UOVF has occurred. Write 1 to clear this status bit. |
|         |            | 1 Interrupt Latched                                                                                                                             |

**Table 30-36:** PVP\_ILAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | THC1RDY  | THC1 Report Ready Latch.<br>The PVP_ILAT.THClRDY bit indicates that the PVP_ILAT.THClRDY has occurred. Write 1 to clear this status bit.           |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 16      | THC0RDY  | THC0 Report Ready Latch.<br>The PVP_ILAT.THc0RDY bit indicates that the PVP_ILAT.THc0RDY has occurred. Write 1 to clear this status bit.           |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 15      | MPRDY    | Memory Pipe Ready Latch.<br>The PVP_ILAT.MPRDY bit indicates that the PVP_ILAT.MPRDY has occurred. Write 1 to clear this status bit.               |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 14      | CPRDY    | Camera Pipe Ready Latch.<br>The PVP_ILAT.CPRDY bit indicates that the PVP_ILAT.CPRDY has occurred. Write 1 to clear this status bit.               |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 13      | MPDRN    | Memory Pipe Drain Done Latch.<br>The PVP_ILAT.MPDRN bit indicates that the PVP_ILAT.MPDRN has occurred. Write 1 to clear this status bit.          |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 12      | CPDRN    | Camera Pipe Drain Done Latch.<br>The PVP_ILAT.CPDRN bit indicates that the PVP_ILAT.CPDRN has occurred. Write 1 to clear this status bit.          |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 10      | CPIPFOVF | Camera Pipe Pixel Overrun Latch.<br>The PVP_ILAT.CPIPFOVF bit indicates that the PVP_ILAT.CPIPFOVF has occurred. Write 1 to clear this status bit. |
|         |          | 1 Interrupt Latched                                                                                                                                |

**Table 30-36:** PVP\_ILAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| 9       | MPOPFDAT | Memory Pipe First Pixel Latch.<br>The PVP_ILAT.MPOPFDAT bit indicates that the PVP_ILAT.MPOPFDAT has occurred. Write 1 to clear this status bit.   |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 8       | CPOPFDAT | Camera Pipe First Pixel Latch.<br>The PVP_ILAT.CPOPFDAT bit indicates that the PVP_ILAT.CPOPFDAT has occurred. Write 1 to clear this status bit.   |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 7       | CPSTOVF  | Status DDE Stall Error Latch.<br>The PVP_ILAT.CPSTOVF bit indicates that the PVP_ILAT.CPSTOVF has occurred. Write 1 to clear this status bit.      |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 6       | OPF2OVF  | OPF2 DDE Stall Error Latch.<br>The PVP_ILAT.OPF2OVF bit indicates that the PVP_ILAT.OPF2OVF has occurred. Write 1 to clear this status bit.        |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 5       | OPF1OVF  | OPF1 DDE Stall Error Latch.<br>The PVP_ILAT.OPF1OVF bit indicates that the PVP_ILAT.OPF1OVF has occurred. Write 1 to clear this status bit.        |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 4       | OPF0OVF  | OPF0 DDE Stall Error Latch.<br>The PVP_ILAT.OPF0OVF bit indicates that the PVP_ILAT.OPF0OVF has occurred. Write 1 to clear this status bit.        |
|         |          | 1 Interrupt Latched                                                                                                                                |
| 3       | MPWRERR  | Memory Pipe MMR Write Error Latch.<br>The PVP_ILAT.MPWRERR bit indicates that the PVP_ILAT.MPWRERR has occurred. Write 1 to clear this status bit. |
|         |          | 1 Interrupt Latched                                                                                                                                |

**Table 30-36:** PVP\_ILAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                      |                   |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 2       | CPWRERR  | Camera Pipe MMR Write Error Latch.<br>The <code>PVP_ILAT.CPWRERR</code> bit indicates that the <code>PVP_ILAT.CPWRERR</code> has occurred. Write 1 to clear this status bit. |                   |
|         |          | 1                                                                                                                                                                            | Interrupt Latched |
| 1       | MPDC     | Memory Pipe DC Mask.<br>The <code>PVP_ILAT.MPDC</code> bit indicates that the <code>PVP_ILAT.MPDC</code> has occurred. Write 1 to clear this status bit.                     |                   |
|         |          | 1                                                                                                                                                                            | Interrupt Latched |
| 0       | CPDC     | Camera Pipe DC Latch.<br>The <code>PVP_ILAT.CPDC</code> bit indicates that the <code>PVP_ILAT.CPDC</code> has occurred. Write 1 to clear this status bit.                    |                   |
|         |          | 1                                                                                                                                                                            | Interrupt Latched |

## Interrupt Request n

The `PVP_IREQn` register contains the latched interrupt request status. These requests reflect the status of interrupts that are latched in the `PVP_ILAT` register and are unmasked (enabled) in the `PVP_IMSKn`. The `PVP_IREQn` bits are set by PVP status, but cleared through a write-1-to-clear operation. Note that clearing a bit in `PVP_IREQn` clears the corresponding bit in the `PVP_ILAT` register.

PVP\_IREQn: Interrupt Request n - R/W

Reset = 0x0000 0000

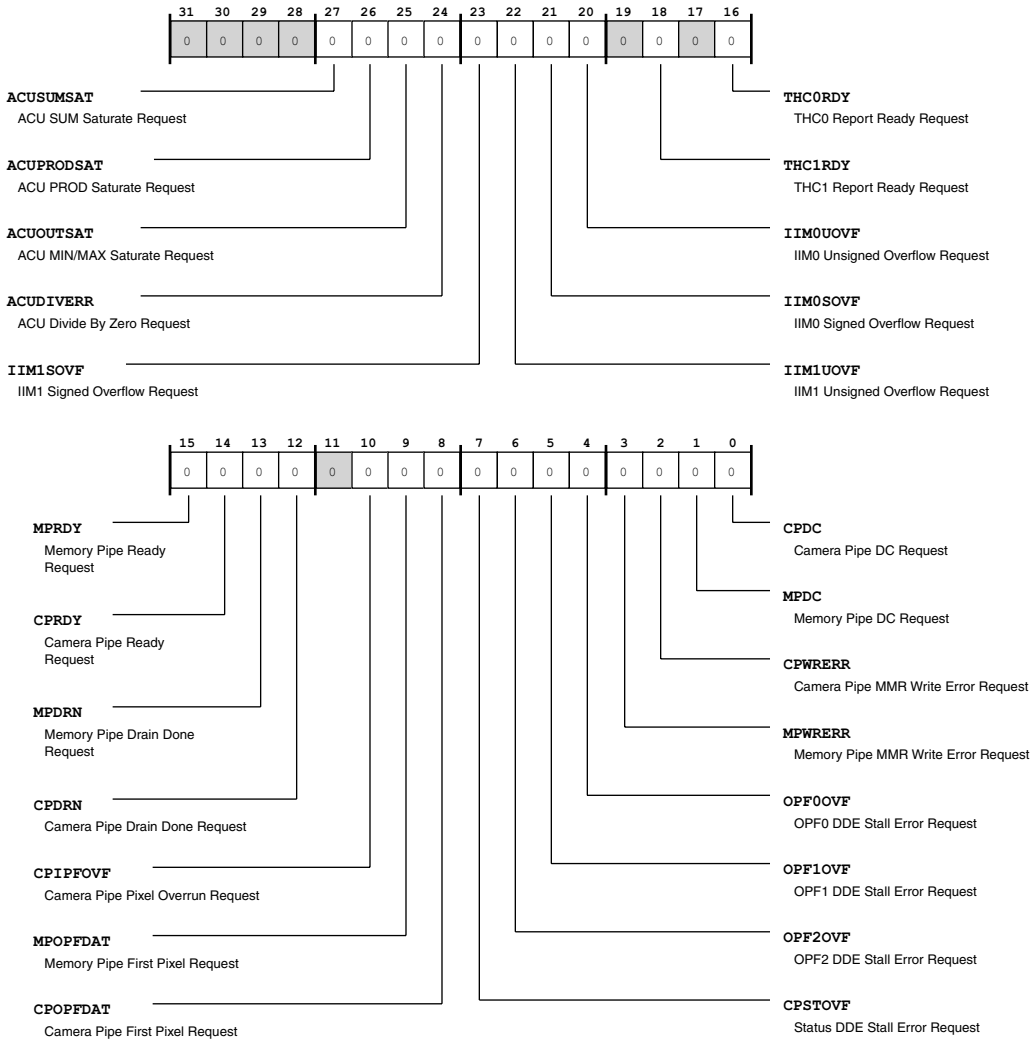


Figure 30-71: PVP\_IREQn Register Diagram

Table 30-37: PVP\_IREQn Register Fields

| Bit No. | Bit Name   | Description/Enumeration    |
|---------|------------|----------------------------|
| 27      | ACUSUMSAT  | ACU SUM Saturate Request.  |
|         |            | 1 Unmasked Int Req Latched |
| 26      | ACUPRODSAT | ACU PROD Saturate Request. |
|         |            | 1 Unmasked Int Req Latched |



**Table 30-37:** PVP\_IREQn Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration         |                          |
|---------|-----------|---------------------------------|--------------------------|
| 25      | ACUOUTSAT | ACU MIN/MAX Saturate Request.   |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 24      | ACUDIVERR | ACU Divide By Zero Request.     |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 23      | IIM1SOVF  | IIM1 Signed Overflow Request.   |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 22      | IIM1UOVF  | IIM1 Unsigned Overflow Request. |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 21      | IIM0SOVF  | IIM0 Signed Overflow Request.   |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 20      | IIM0UOVF  | IIM0 Unsigned Overflow Request. |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 18      | THC1RDY   | THC1 Report Ready Request.      |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 16      | THC0RDY   | THC0 Report Ready Request.      |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 15      | MPRDY     | Memory Pipe Ready Request.      |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 14      | CPRDY     | Camera Pipe Ready Request.      |                          |
|         |           | 1                               | Unmasked Int Req Latched |
| 13      | MPDRN     | Memory Pipe Drain Done Request. |                          |
|         |           | 1                               | Unmasked Int Req Latched |

**Table 30-37:** PVP\_IREQn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration              |                          |
|---------|----------|--------------------------------------|--------------------------|
| 12      | CPDRN    | Camera Pipe Drain Done Request.      |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 10      | CPIPFOVF | Camera Pipe Pixel Overrun Request.   |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 9       | MPOPFDAT | Memory Pipe First Pixel Request.     |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 8       | CPOPFDAT | Camera Pipe First Pixel Request.     |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 7       | CPSTOVF  | Status DDE Stall Error Request.      |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 6       | OPF2OVF  | OPF2 DDE Stall Error Request.        |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 5       | OPF1OVF  | OPF1 DDE Stall Error Request.        |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 4       | OPF0OVF  | OPF0 DDE Stall Error Request.        |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 3       | MPWRERR  | Memory Pipe MMR Write Error Request. |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 2       | CPWRERR  | Camera Pipe MMR Write Error Request. |                          |
|         |          | 1                                    | Unmasked Int Req Latched |
| 1       | MPDC     | Memory Pipe DC Request.              |                          |
|         |          | 1                                    | Unmasked Int Req Latched |

**Table 30-37:** PVP\_IREQn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 0       | CPDC     | Camera Pipe DC Request.    |
|         |          | 1 Unmasked Int Req Latched |

OPFn (Camera Pipe) Configuration

The output data formatter (OPF) blocks house FIFOs that buffer data for a sustained peak bandwidth output. The PVP\_OPFn\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

**PVP\_OPFn\_CFG: OPFn (Camera Pipe) Configuration - R/W**

Reset = 0x0000 0000

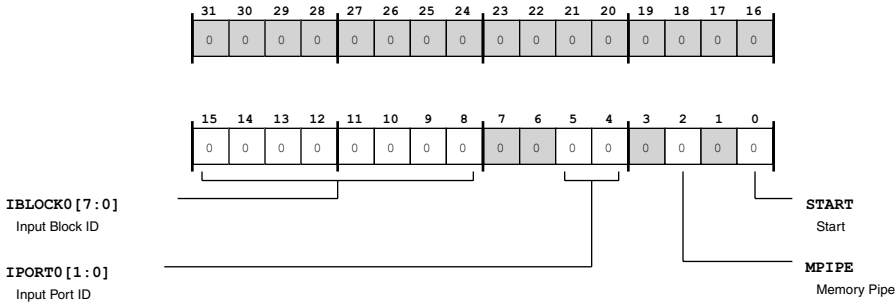


**Figure 30-72:** PVP\_OPFn\_CFG Register Diagram

**Table 30-38:** PVP\_OPFn\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_OPFn_CFG.IBLOCK0 and PVP_OPFn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 5 PEC                                                                                                                                                                                                                                                                                                                                  |
|         |          | 6 IIM0                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 IIM1                                                                                                                                                                                                                                                                                                                                 |
|         |          | 8 ACU                                                                                                                                                                                                                                                                                                                                  |
|         |          | 12 IPF0                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
|         |          | 32 THC0                                                                                                                                                                                                                                                                                                                                |
|         |          | 40 THC1                                                                                                                                                                                                                                                                                                                                |
|         |          | 48 PMA                                                                                                                                                                                                                                                                                                                                 |
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_OPFn_CFG.IBLOCK0 and PVP_OPFn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_OPFn_CFG.MPIPE bit always reads as 0 (camera pipe).                                                                                                                                                                                                                                                            |

**Table 30-38:** PVP\_OPFn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------|
| 0       | START    | Start.<br>The PVP_OPFn_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |
|         |          | 1      Enable (W1A)                                                                           |

## OPFn (Camera Pipe) Control

The PVP\_OPFn\_CTL register controls the output data formatter (OPF) block's pipeline features, including input and output data size.

**PVP\_OPFn\_CTL: OPFn (Camera Pipe) Control - R/W**

Reset = 0x0000 0000

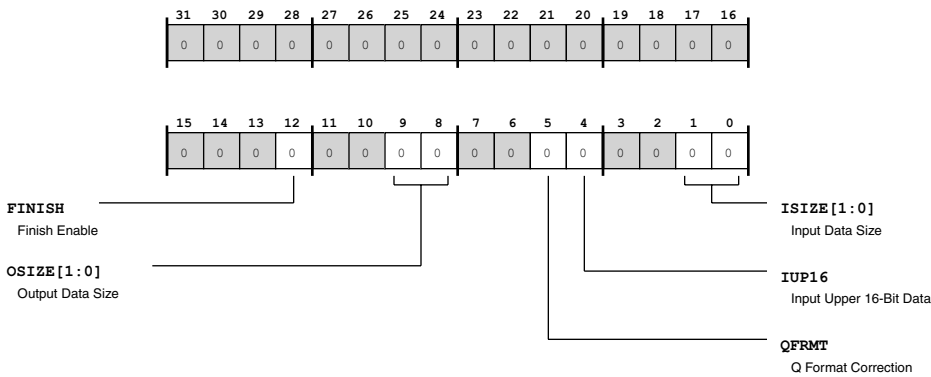


**Figure 30-73:** PVP\_OPFn\_CTL Register Diagram

**Table 30-39:** PVP\_OPFn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 12      | FINISH   | Finish Enable.<br>The PVP_OPFn_CTL.FINISH enables the PVP to signal finished on frame boundary. |
|         |          | 0      Disable Finish Signal                                                                    |
|         |          | 1      Enable Finish Signal                                                                     |

**Table 30-39:** PVP\_OPFn\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | OSIZE    | Output Data Size.<br>The PVP_OPFn_CTL.OSIZE selects the output data size. Note that the PVP_OPFn_CTL.OSIZE selection should be equal to or more than that of PVP_OPFn_CTL.ISIZE.                                                                                                            |
|         |          | 0 32-Bit Output Data Size                                                                                                                                                                                                                                                                   |
|         |          | 1 16-Bit Output Data Size                                                                                                                                                                                                                                                                   |
|         |          | 2 8-Bit Output Data Size                                                                                                                                                                                                                                                                    |
| 5       | QFRMT    | Q Format Correction.<br>The PVP_OPFn_CTL.QFRMT enables Q format correction, converting data to unsigned by 1 left shift.                                                                                                                                                                    |
|         |          | 0 Disable Q Format Correction                                                                                                                                                                                                                                                               |
|         |          | 1 Enable Q Format Correction                                                                                                                                                                                                                                                                |
| 4       | IUP16    | Input Upper 16-Bit Data.<br>The PVP_OPFn_CTL.IUP16 selects whether the input data word is the upper or lower 16 bits (for input data sizes of less than 32 bits). Note that for PVP_OPFn_CTL.ISIZE of 8 bit or 4 bit, the input data is in the least significant bits of the lower 16 bits. |
|         |          | 0 Lower 16 Bits                                                                                                                                                                                                                                                                             |
|         |          | 1 Upper 16 Bits                                                                                                                                                                                                                                                                             |
| 1:0     | ISIZE    | Input Data Size.<br>The PVP_OPFn_CTL.ISIZE selects the input data size.                                                                                                                                                                                                                     |
|         |          | 0 32-Bit Input Data Size                                                                                                                                                                                                                                                                    |
|         |          | 1 16-Bit Input Data Size                                                                                                                                                                                                                                                                    |
|         |          | 2 8-Bit Input Data Size                                                                                                                                                                                                                                                                     |
|         |          | 3 4-Bit Input Data Size                                                                                                                                                                                                                                                                     |

# OPF3 (Memory Pipe) Configuration

The output data formatter (OPF) blocks house FIFOs that buffer data for a sustained peak bandwidth output. The PVP\_OPF3\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_OPF3\_CFG: OPF3 (Memory Pipe) Configuration - R/W

Reset = 0x0000 0004

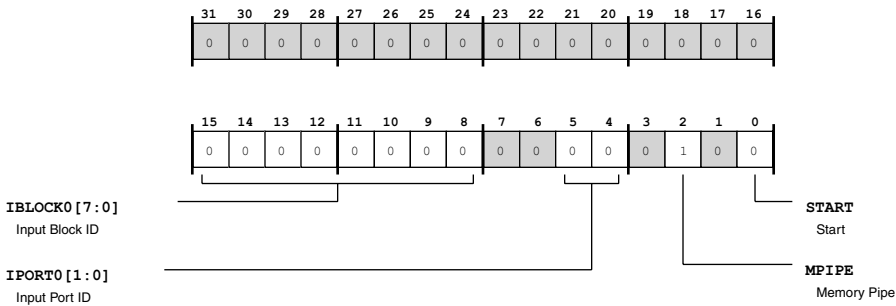


Figure 30-74: PVP\_OPF3\_CFG Register Diagram

**Table 30-40:** PVP\_OPF3\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_OPF3_CFG.IBLOCK0 and PVP_OPF3_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 5 PEC                                                                                                                                                                                                                                                                                                                                  |
|         |          | 6 IIM0                                                                                                                                                                                                                                                                                                                                 |
|         |          | 7 IIM1                                                                                                                                                                                                                                                                                                                                 |
|         |          | 8 ACU                                                                                                                                                                                                                                                                                                                                  |
|         |          | 10 UDS                                                                                                                                                                                                                                                                                                                                 |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
|         |          | 32 THC0                                                                                                                                                                                                                                                                                                                                |
|         |          | 40 THC1                                                                                                                                                                                                                                                                                                                                |
|         |          | 48 PMA                                                                                                                                                                                                                                                                                                                                 |
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_OPF3_CFG.IBLOCK0 and PVP_OPF3_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_OPF3_CFG.MPIPE bit always reads as 1 (memory pipe).                                                                                                                                                                                                                                                            |



**Table 30-40:** PVP\_OPF3\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------|
| 0       | START    | Start.<br>The PVP_OPF3_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |
|         |          | 1<br>Enable (W1A)                                                                             |

### OPF3 (Memory Pipe) Control

The PVP\_OPF3\_CTL register controls the OPF3 block's pipeline features, including input and output data size.

**PVP\_OPF3\_CTL: OPF3 (Memory Pipe) Control - R/W**

Reset = 0x0000 0000

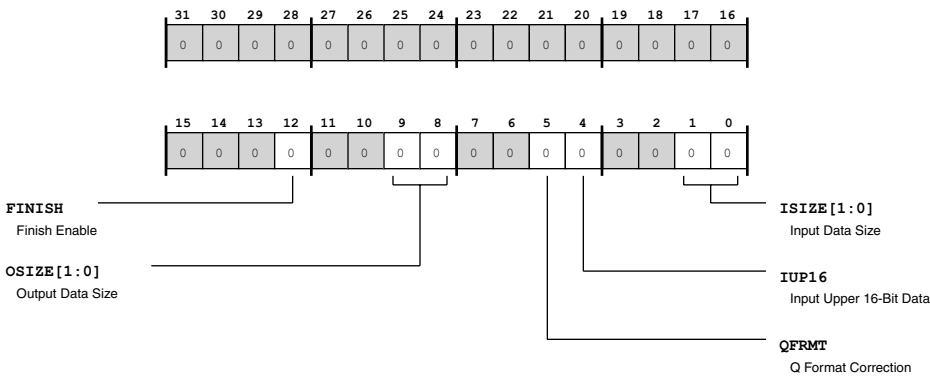


**Figure 30-75:** PVP\_OPF3\_CTL Register Diagram

**Table 30-41:** PVP\_OPF3\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------|
| 12      | FINISH   | Finish Enable.<br>The PVP_OPF3_CTL.FINISH enables the PVP to signal finished on frame boundary. |
|         |          | 0<br>Disable Finish Signal                                                                      |
|         |          | 1<br>Enable Finish Signal                                                                       |

**Table 30-41:** PVP\_OPF3\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | OSIZE    | Output Data Size.<br>The PVP_OPF3_CTL.OSIZE selects the output data size. Note that the PVP_OPF3_CTL.OSIZE selection should be equal to or more than that of PVP_OPF3_CTL.ISIZE.                                                                                                            |
|         |          | 0 32-Bit Output Data Size                                                                                                                                                                                                                                                                   |
|         |          | 1 16-Bit Output Data Size                                                                                                                                                                                                                                                                   |
|         |          | 2 8-Bit Output Data Size                                                                                                                                                                                                                                                                    |
| 5       | QFRMT    | Q Format Correction.<br>The PVP_OPF3_CTL.QFRMT enables Q format correction, converting data to unsigned by 1 left shift.                                                                                                                                                                    |
|         |          | 0 Disable Q Format Correction                                                                                                                                                                                                                                                               |
|         |          | 1 Enable Q Format Correction                                                                                                                                                                                                                                                                |
| 4       | IUP16    | Input Upper 16-Bit Data.<br>The PVP_OPF3_CTL.IUP16 selects whether the input data word is the upper or lower 16 bits (for input data sizes of less than 32 bits). Note that for PVP_OPF3_CTL.ISIZE of 8 bit or 4 bit, the input data is in the least significant bits of the lower 16 bits. |
|         |          | 0 Lower 16 Bits                                                                                                                                                                                                                                                                             |
|         |          | 1 Upper 16 Bits                                                                                                                                                                                                                                                                             |
| 1:0     | ISIZE    | Input Data Size.<br>The PVP_OPF3_CTL.ISIZE selects the input data size.                                                                                                                                                                                                                     |
|         |          | 0 32-Bit Input Data Size                                                                                                                                                                                                                                                                    |
|         |          | 1 16-Bit Input Data Size                                                                                                                                                                                                                                                                    |
|         |          | 2 8-Bit Input Data Size                                                                                                                                                                                                                                                                     |
|         |          | 3 4-Bit Input Data Size                                                                                                                                                                                                                                                                     |

PEC Configuration

The pixel edge classifier (PEC) block performs pre-processing for the first derivative canny edge detection algorithm. The PVP\_PEC\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_PEC\_CFG: PEC Configuration - R/W

Reset = 0x0000 0000

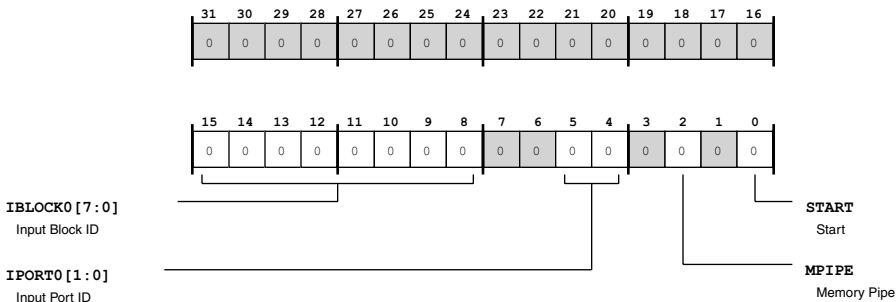


Figure 30-76: PVP\_PEC\_CFG Register Diagram

Table 30-42: PVP\_PEC\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PEC_CFG.IBLOCK0 and PVP_PEC_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). |
|         |          | 8 ACU                                                                                                                                                                                                                   |
|         |          | 12 IPF0                                                                                                                                                                                                                 |
|         |          | 14 IPF1                                                                                                                                                                                                                 |
|         |          | 16 CNV0                                                                                                                                                                                                                 |
|         |          | 20 CNV1                                                                                                                                                                                                                 |
|         |          | 24 CNV2                                                                                                                                                                                                                 |
|         |          | 28 CNV3                                                                                                                                                                                                                 |
|         |          | 48 PMA                                                                                                                                                                                                                  |

Table 30-42: PVP\_PEC\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PEC_CFG.IBLOCK0 and PVP_PEC_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_PEC_CFG.MPIPE bit selects memory pipe or camera pipe.                                                                                                                                                                                                                                                        |
|         |          | 0                      Camera Pipe                                                                                                                                                                                                                                                                                                   |
|         |          | 1                      Memory Pipe                                                                                                                                                                                                                                                                                                   |
| 0       | START    | Start.<br>The PVP_PEC_CFG.START enables the PVP. This bit reads 0 after the pipe is drained.                                                                                                                                                                                                                                         |
|         |          | 1                      Enable (W1A)                                                                                                                                                                                                                                                                                                  |

PEC Control

The PVP\_PEC\_CTL register selects operation modes for the pixel edge classifier (PEC) block, including 1st derivative mode (PEC1) and 2nd derivative mode (PEC2).

PVP\_PEC\_CTL: PEC Control - R/W

Reset = 0x0000 0000

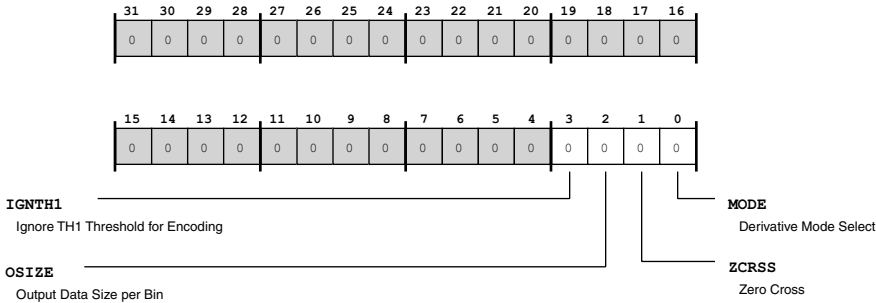


Figure 30-77: PVP\_PEC\_CTL Register Diagram

**Table 30-43:** PVP\_PEC\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | IGNTH1   | Ignore TH1 Threshold for Encoding.<br>The PVP_PEC_CTL.IGNTH1 selects PEC encoding control for strong and weak edges. This bit is valid in both 1st derivative and 2nd derivative modes of PEC.                                                                                                                                                      |
|         |          | 0 Different Strong/Weak Edge Encoding                                                                                                                                                                                                                                                                                                               |
|         |          | 1 Identical Strong/Weak Edge Encoding                                                                                                                                                                                                                                                                                                               |
| 2       | OSIZE    | Output Data Size per Bin.<br>The PVP_PEC_CTL.OSIZE selects the output data size per bin. This bit is valid in both 1st derivative and 2nd derivative modes of PEC.                                                                                                                                                                                  |
|         |          | 0 8 Bits Per Bin PEC Output Data Size                                                                                                                                                                                                                                                                                                               |
|         |          | 1 16 Bits Per Bin PEC Output Data Size                                                                                                                                                                                                                                                                                                              |
| 1       | ZCRSS    | Zero Cross.<br>The PVP_PEC_CTL.ZCRSS selects the sub-pixel value output to include sub-pixel values and either zero crossing codes or classified angle indices.                                                                                                                                                                                     |
|         |          | 0 Angle Indices and Sub-Pixel Values                                                                                                                                                                                                                                                                                                                |
|         |          | 1 Zero Crossing Codes and Sub-Pixel Values                                                                                                                                                                                                                                                                                                          |
| 0       | MODE     | Derivative Mode Select.<br>The PVP_PEC_CTL.MODE selects that derivative mode for the PEC. Note that the PVP_PEC_CTL.MODE bits only can be changed when the PVP_IPFn_PIECTL.DRAIN bit is set. If set, the PVP works on 16-bit, signed, 2nd derivative inputs. If cleared, the PVP works on 16-bit, unsigned, gradient magnitude and 5-bit angle bin. |
|         |          | 0 1st Derivative Mode                                                                                                                                                                                                                                                                                                                               |
|         |          | 1 2nd Derivative Mode                                                                                                                                                                                                                                                                                                                               |

## PEC Lower Hysteresis Threshold

The PVP\_PEC\_D1TH0 provides the lower threshold value used by the Canny algorithm's hysteresis thresholding for streaking elimination.

PVP\_PEC\_D1TH0: PEC Lower Hysteresis Threshold - R/W

Reset = 0x0000 0000

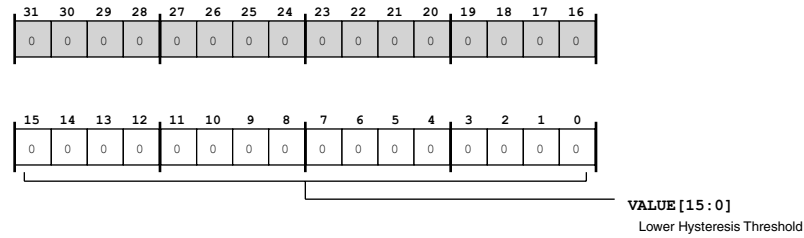


Figure 30-78: PVP\_PEC\_D1TH0 Register Diagram

Table 30-44: PVP\_PEC\_D1TH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | VALUE    | Lower Hysteresis Threshold. |

PEC Upper Hysteresis Threshold

The PVP\_PEC\_D1TH1 provides the upper threshold value used by the Canny algorithm's hysteresis thresholding for streaking elimination.

PVP\_PEC\_D1TH1: PEC Upper Hysteresis Threshold - R/W

Reset = 0x0000 0000

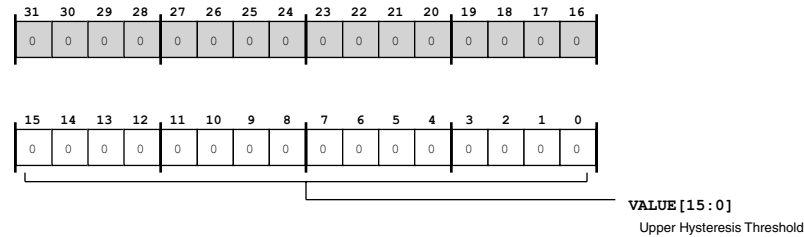


Figure 30-79: PVP\_PEC\_D1TH1 Register Diagram

Table 30-45: PVP\_PEC\_D1TH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 15:0    | VALUE    | Upper Hysteresis Threshold. |

## PEC Weak Zero Crossing Threshold

The PVP\_PEC\_D2TH0 provides an upper threshold that the PEC uses to remove the noise during weak zero crossing detection. Zero crossings are considered valid only if there is a sign change and id the difference between the 2nd derivative values of these two pixels is greater than the threshold.

PVP\_PEC\_D2TH0: PEC Weak Zero Crossing Threshold - R/W

Reset = 0x0000 0000

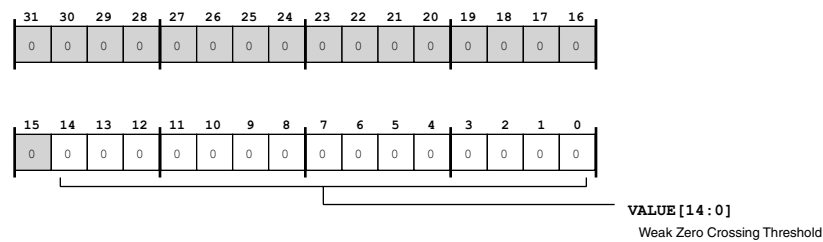


Figure 30-80: PVP\_PEC\_D2TH0 Register Diagram

Table 30-46: PVP\_PEC\_D2TH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 14:0    | VALUE    | Weak Zero Crossing Threshold. |

## PEC Strong Zero Crossing Threshold

The PVP\_PEC\_D2TH1 provides an upper threshold that the PEC uses to remove the noise during strong zero crossing detection. Zero crossings are considered valid only if there is a sign change and id the difference between the 2nd derivative values of these two pixels is greater than the threshold.

PVP\_PEC\_D2TH1: PEC Strong Zero Crossing Threshold - R/W

Reset = 0x0000 0000

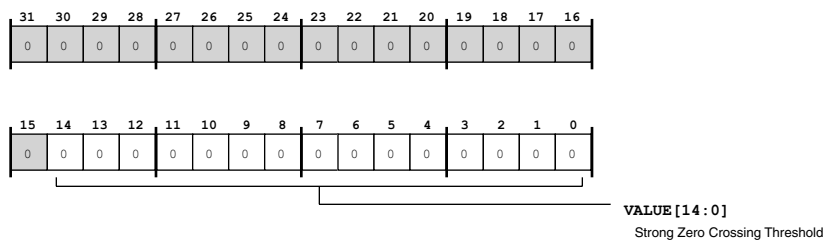


Figure 30-81: PVP\_PEC\_D2TH1 Register Diagram

Table 30-47: PVP\_PEC\_D2TH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 14:0    | VALUE    | Strong Zero Crossing Threshold. |

IIMn Configuration

The integral image computation (IIM) block computes the rectangular integral image algorithm. The PVP\_IIMn\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_IIMn\_CFG: IIMn Configuration - R/W

Reset = 0x0000 0000

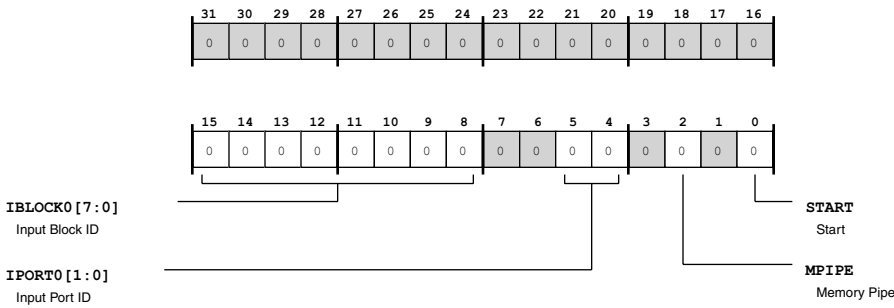


Figure 30-82: PVP\_IIMn\_CFG Register Diagram



**Table 30-48:** PVP\_IIMn\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_IIMn_CFG.IBLOCK0 and PVP_IIMn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 5 PEC                                                                                                                                                                                                                                                                                                                                  |
|         |          | 8 ACU                                                                                                                                                                                                                                                                                                                                  |
|         |          | 12 IPF0                                                                                                                                                                                                                                                                                                                                |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
|         |          | 32 THC0                                                                                                                                                                                                                                                                                                                                |
|         |          | 40 THC1                                                                                                                                                                                                                                                                                                                                |
|         |          | 48 PMA                                                                                                                                                                                                                                                                                                                                 |
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_IIMn_CFG.IBLOCK0 and PVP_IIMn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_IIMn_CFG.MPIPE bit selects memory pipe or camera pipe.                                                                                                                                                                                                                                                         |
|         |          | 0 Camera Pipe                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Memory Pipe                                                                                                                                                                                                                                                                                                                          |

Table 30-48: PVP\_IIMn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |              |
|---------|----------|-----------------------------------------------------------------------------------------------|--------------|
| 0       | START    | Start.<br>The PVP_IIMn_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |              |
|         |          | 1                                                                                             | Enable (W1A) |

IIMn Control

The PVP\_IIMn\_CTL register selects operation modes for the integral image computation (IIM) block, including rectangular mode, diagonal mode, and row mode.

PVP\_IIMn\_CTL: IIMn Control - R/W

Reset = 0x0000 0000

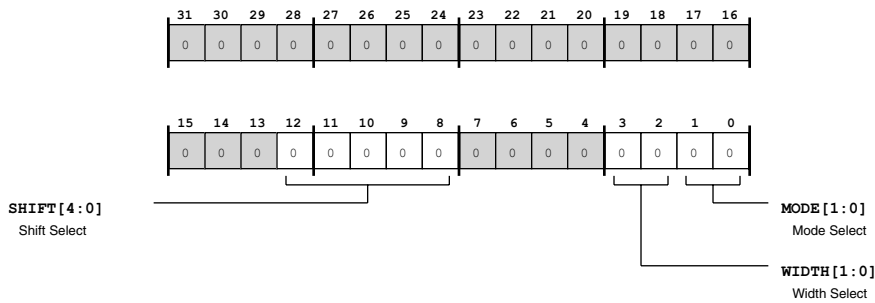


Figure 30-83: PVP\_IIMn\_CTL Register Diagram

Table 30-49: PVP\_IIMn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------|
| 12:8    | SHIFT    | Shift Select.<br>The PVP_IIMn_CTL.SHIFT selects the amount of arithmetic right shift on input from 0 to 31 bits. |

**Table 30-49:** PVP\_IIMn\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3:2     | WIDTH    | Width Select.<br>The PVP_IIMn_CTL.WIDTH selects the input data width for rectangular mode (SAT). This field is applicable only when PVP_IIMn_CTL.MODE =00 and is invalid for other IIM modes. |
|         |          | 0 Single 32 Bit                                                                                                                                                                               |
|         |          | 1 Dual 16 Bit                                                                                                                                                                                 |
|         |          | 2 Reserved                                                                                                                                                                                    |
|         |          | 3 Quad 8 Bit                                                                                                                                                                                  |
| 1:0     | MODE     | Mode Select.<br>The PVP_IIMn_CTL.MODE selects the IIM mode.                                                                                                                                   |
|         |          | 0 Rectangular Mode ( SAT)                                                                                                                                                                     |
|         |          | 1 Diagonal Mode ( RSAT -45)                                                                                                                                                                   |
|         |          | 2 Row Mode                                                                                                                                                                                    |
|         |          | 3 Reserved                                                                                                                                                                                    |

## IIMn Scaling Values

The PVP\_IIMn\_SCALE holds the independently controlled scaling factors in each dimension for the output's programmable downscaler.

### PVP\_IIMn\_SCALE: IIMn Scaling Values - R/W

Reset = 0x0000 0000

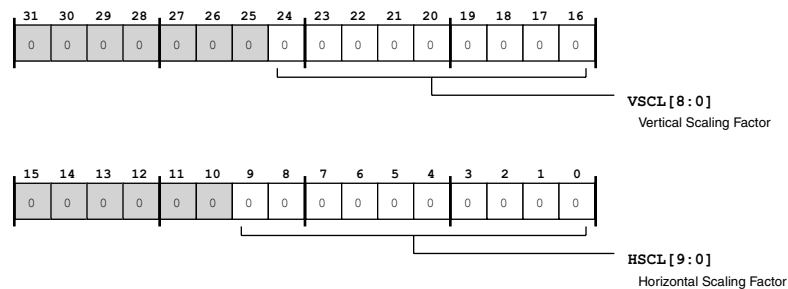


**Figure 30-84:** PVP\_IIMn\_SCALE Register Diagram

Table 30-50: PVP\_IIMn\_SCALE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                               |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24:16   | VSCL     | Vertical Scaling Factor.<br>The PVP_IIMn_SCALE.VSCL holds the vertical scaling factor, which can be selected to be any power of 2 from 1 to 512.      |
| 9:0     | HSCL     | Horizontal Scaling Factor.<br>The PVP_IIMn_SCALE.HSCL holds the horizontal scaling factor, which can be selected to be any power of 2 from 1 to 1024. |

IIMn Signed Overflow Status

The PVP\_IIMn\_SOVF\_STAT holds the signed overflow status, which is determined considering addition of 32-bit signed numbers.

PVP\_IIMn\_SOVF\_STAT: IIMn Signed Overflow Status - R/NW

Reset = 0x03ff 07ff

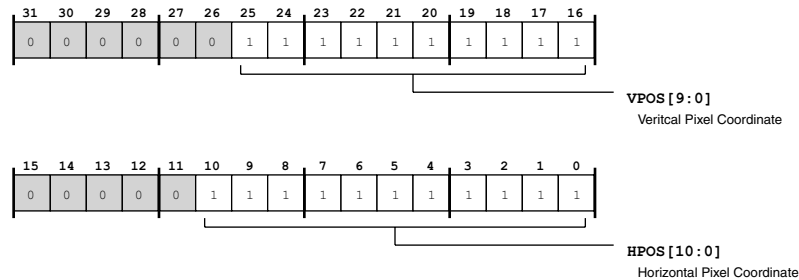


Figure 30-85: PVP\_IIMn\_SOVF\_STAT Register Diagram

Table 30-51: PVP\_IIMn\_SOVF\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 25:16   | VPOS     | Veritcal Pixel Coordinate.<br>The PVP_IIMn_SOVF_STAT.VPOS holds the bertical coordinate of the pixel which resulted in signed saturation.     |
| 10:0    | HPOS     | Horizontal Pixel Coordinate.<br>The PVP_IIMn_SOVF_STAT.HPOS holds the horizontal coordinate of the pixel which resulted in signed saturation. |

# IIMn Unsigned Overflow Status

The PVP\_IIMn\_UOVF\_STAT holds the unsigned overflow status, which is determined considering addition of 32-bit unsigned numbers.

PVP\_IIMn\_UOVF\_STAT: IIMn Unsigned Overflow Status - R/NW

Reset = 0x03ff 07ff

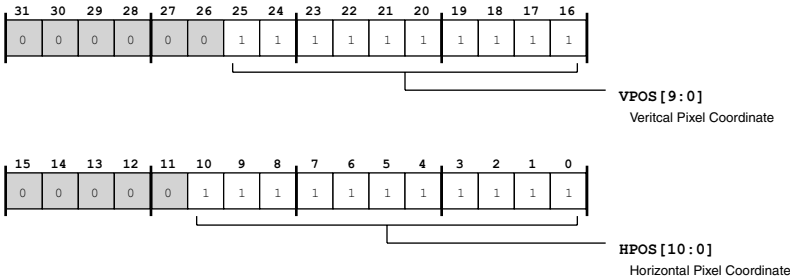


**Figure 30-86:** PVP\_IIMn\_UOVF\_STAT Register Diagram

**Table 30-52:** PVP\_IIMn\_UOVF\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                         |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| 25:16   | VPOS     | Veritcal Pixel Coordinate.<br>The PVP_IIMn_UOVF_STAT.VPOS holds the vertical coordinate of the pixel which resulted in unsigned saturation.     |
| 10:0    | HPOS     | Horizontal Pixel Coordinate.<br>The PVP_IIMn_UOVF_STAT.HPOS holds the horizontal coordinate of the pixel which resulted in unsigned saturation. |

# ACU Configuration

The arithmetic computation unit (ACU) block provides basic 32-bit math such as addition, subtraction, multiplication, and division. This block also features a 48-bit accumulator and can normalize and saturate on the output. The PVP\_ACU\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_ACU\_CFG: ACU Configuration - R/W

Reset = 0x0000 0000

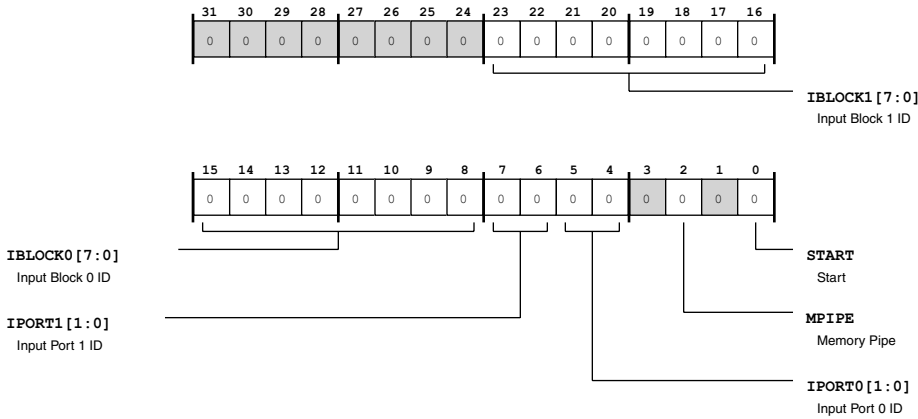


Figure 30-87: PVP\_ACU\_CFG Register Diagram

Table 30-53: PVP\_ACU\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:16   | IBLOCK1  | Input Block 1 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_ACU_CFG.IBLOCK1 and PVP_ACU_CFG.IPORT1 fields determine which output port(s) are connected to the input port(s). |
|         |          | 12 IPF0                                                                                                                                                                                                                   |
|         |          | 14 IPF1                                                                                                                                                                                                                   |
|         |          | 16 CNV0                                                                                                                                                                                                                   |
|         |          | 20 CNV1                                                                                                                                                                                                                   |
|         |          | 24 CNV2                                                                                                                                                                                                                   |
|         |          | 28 CNV3                                                                                                                                                                                                                   |

**Table 30-53:** PVP\_ACU\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block 0 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_ACU_CFG.IBLOCK0 and PVP_ACU_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 12 IPF0                                                                                                                                                                                                                                                                                                                                |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
| 7:6     | IPOINT1  | Input Port 1 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_ACU_CFG.IBLOCK1 and PVP_ACU_CFG.IPORT1 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 5:4     | IPOINT0  | Input Port 0 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_ACU_CFG.IBLOCK0 and PVP_ACU_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_ACU_CFG.MPIPE bit selects memory pipe or camera pipe.                                                                                                                                                                                                                                                          |
|         |          | 0 Camera Pipe                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Memory Pipe                                                                                                                                                                                                                                                                                                                          |

Table 30-53: PVP\_ACU\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------|
| 0       | START    | Start.<br>The PVP_ACU_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |
|         |          | 1<br>Write to Enable the Module<br>Enable (W1A)                                              |

ACU Control

The PVP\_ACU\_CTL register selects operation modes for the arithmetic computation unit (ACU), including modes for sum, product, accumulate, and shift operations.

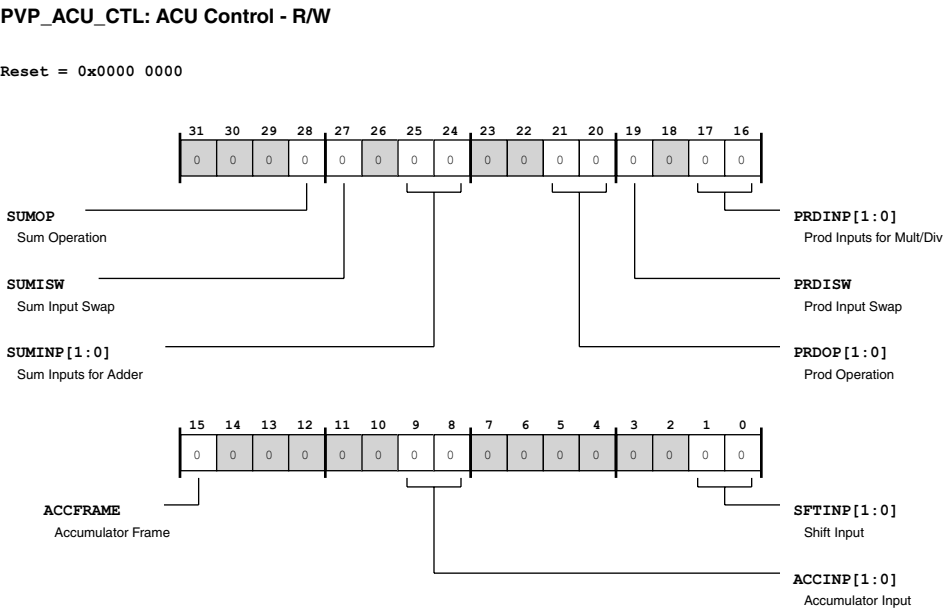


Figure 30-88: PVP\_ACU\_CTL Register Diagram



**Table 30-54:** PVP\_ACU\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 28      | SUMOP    | Sum Operation.<br>The PVP_ACU_CTL.SUMOP selects add or subtract as the SUM function.                                                         |
|         |          | 0 Add                                                                                                                                        |
|         |          | 1 Subtract                                                                                                                                   |
| 27      | SUMISW   | Sum Input Swap.<br>The PVP_ACU_CTL.SUMISW selects swapping of operands for the SUM function. The swap is meaningful for subtract operations. |
|         |          | 0 Do Not Swap Operands                                                                                                                       |
|         |          | 1 Swap Operands                                                                                                                              |
| 25:24   | SUMINP   | Sum Inputs for Adder.<br>The PVP_ACU_CTL.SUMINP selects the inputs to the adder for the SUM function.                                        |
|         |          | 0 X,Y Inputs                                                                                                                                 |
|         |          | 1 X,OFFSET Inputs                                                                                                                            |
|         |          | 2 Y,OFFSET Inputs                                                                                                                            |
|         |          | 3 Reserved                                                                                                                                   |
| 21:20   | PRDOP    | Prod Operation.<br>The PVP_ACU_CTL.PRDOP selects multiply or divide as the PROD operation.                                                   |
|         |          | 0 Multiply                                                                                                                                   |
|         |          | 1 Divide with Quotient                                                                                                                       |
|         |          | 2 Divide with Modulus                                                                                                                        |
|         |          | 3 Reserved                                                                                                                                   |

**Table 30-54:** PVP\_ACU\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 19      | PRDISW   | Prod Input Swap.<br>The PVP_ACU_CTL.PRDISW selects swapping of operands for the PROD function. The swap is meaningful for division operations. |
|         |          | 0 Do Not Swap Operands                                                                                                                         |
|         |          | 1 Swap Operands                                                                                                                                |
| 17:16   | PRDINP   | Prod Inputs for Mult/Div.<br>The PVP_ACU_CTL.PRDINP selects the inputs to the multiplier/divider for the PROD function.                        |
|         |          | 0 X,Y Inputs                                                                                                                                   |
|         |          | 1 X,FACTOR Inputs                                                                                                                              |
|         |          | 2 Y,FACTOR Inputs                                                                                                                              |
|         |          | 3 SUM,FACTOR Inputs                                                                                                                            |
| 15      | ACCFRAME | Accumulator Frame.<br>The PVP_ACU_CTL.ACCFRAME selects whether or not the accumulator is output and cleared after each row or frame.           |
|         |          | 0 Clear ACC After Row                                                                                                                          |
|         |          | 1 Clear ACC After Frame                                                                                                                        |
| 9:8     | ACCINP   | Accumulator Input.<br>The PVP_ACU_CTL.ACCINP selects the input to the accumulator for accumulate operations.                                   |
|         |          | 0 X Input                                                                                                                                      |
|         |          | 1 SUM Input                                                                                                                                    |
|         |          | 2 PROD Input                                                                                                                                   |
|         |          | 3 Reserved                                                                                                                                     |

**Table 30-54:** PVP\_ACU\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------|
| 1:0     | SFTINP   | Shift Input.<br>The PVP_ACU_CTL.SFTINP selects the input to the shifter for shift operations. |
|         |          | 0 X Input                                                                                     |
|         |          | 1 SUM Result Input                                                                            |
|         |          | 2 PROD Result Input                                                                           |
|         |          | 3 ACC Result Input                                                                            |

## ACU SUM Constant

The PVP\_ACU\_OFFSET provides the constant value (offset) input for the ACU SUM operation when the PVP\_ACU\_CTL.SUMINP field selects the offset as one of the inputs for the operation.

**PVP\_ACU\_OFFSET: ACU SUM Constant - R/W**

Reset = 0x0000 0000

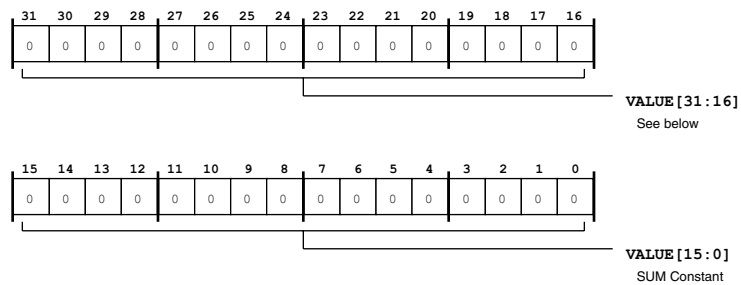


**Figure 30-89:** PVP\_ACU\_OFFSET Register Diagram

**Table 30-55:** PVP\_ACU\_OFFSET Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | SUM Constant.           |

## ACU PROD Constant

The PVP\_ACU\_FACTOR provides the constant value (factor) input for the ACU PROD operation when the PVP\_ACU\_CTL.PR DINP field selects the factor as one of the inputs for the operation.

PVP\_ACU\_FACTOR: ACU PROD Constant - R/W

Reset = 0x0000 0000

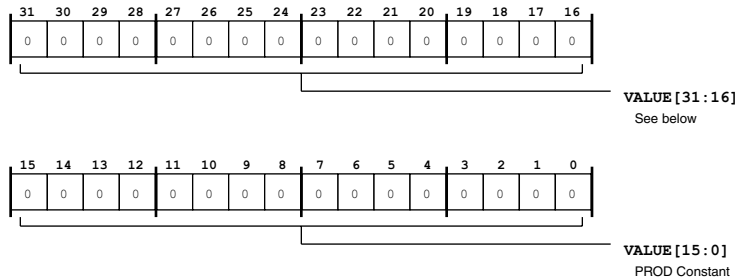


Figure 30-90: PVP\_ACU\_FACTOR Register Diagram

Table 30-56: PVP\_ACU\_FACTOR Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | PROD Constant.          |

ACU Shift Constant

The PVP\_ACU\_SHIFT provides the shift constant value (arithmetic right shift size in bits) for ACC SUM or PROD operations or for 48-bit output from the accumulator. The shift size is from 0 to 31 bits and can be bypassed by programming a value of 0 for the shift constant.

PVP\_ACU\_SHIFT: ACU Shift Constant - R/W

Reset = 0x0000 0000

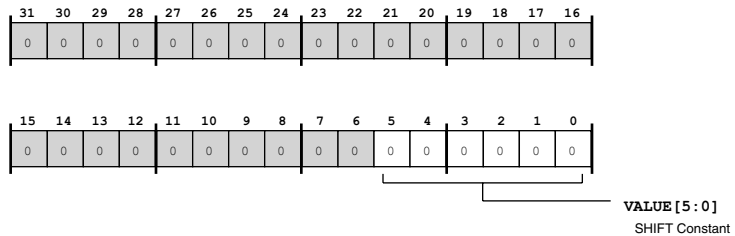


Figure 30-91: PVP\_ACU\_SHIFT Register Diagram

Table 30-57: PVP\_ACU\_SHIFT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 5:0     | VALUE    | SHIFT Constant.         |

# ACU Lower Sat Threshold Min

The PVP\_ACU\_MIN holds the lower threshold value for saturation of ACU shifter output.

**PVP\_ACU\_MIN: ACU Lower Sat Threshold Min - R/W**

Reset = 0x8000 0000

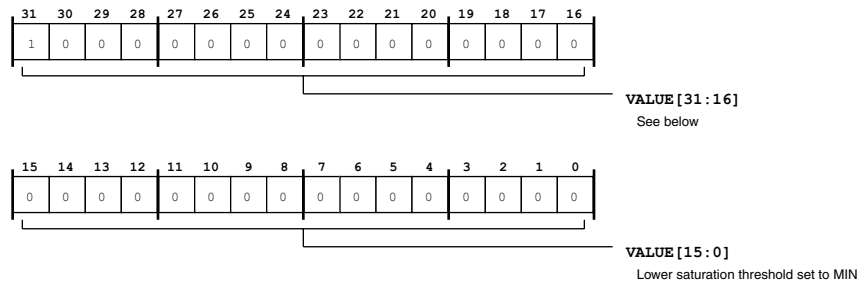


**Figure 30-92:** PVP\_ACU\_MIN Register Diagram

**Table 30-58:** PVP\_ACU\_MIN Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 31:0    | VALUE    | Lower saturation threshold set to MIN. |

# ACU Upper Sat Threshold Max

The PVP\_ACU\_MAX holds the upper threshold value for saturation of ACU shifter output.

**PVP\_ACU\_MAX: ACU Upper Sat Threshold Max - R/W**

Reset = 0x7fff ffff

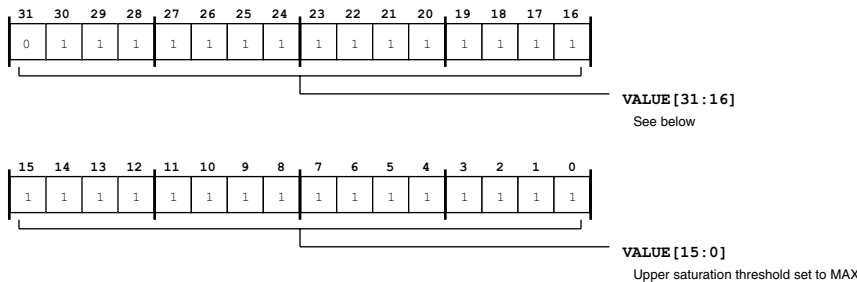


**Figure 30-93:** PVP\_ACU\_MAX Register Diagram

Table 30-59: PVP\_ACU\_MAX Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 31:0    | VALUE    | Upper saturation threshold set to MAX. |

UDS Configuration

The up-down scaler (UDS) block provides image resizing. The PVP\_UDS\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_UDS\_CFG: UDS Configuration - R/W

Reset = 0x0000 0004

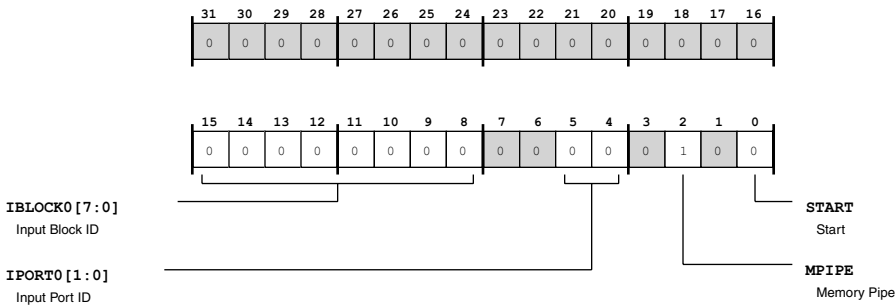


Figure 30-94: PVP\_UDS\_CFG Register Diagram

Table 30-60: PVP\_UDS\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_UDS_CFG.IBLOCK0 and PVP_UDS_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                              |
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_UDS_CFG.IBLOCK0 and PVP_UDS_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |

**Table 30-60:** PVP\_UDS\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------|
| 2       | MPIPE    | Memory Pipe.<br>The PVP_UDS_CFG.MPIPE bit always reads as 1 (memory pipe).                   |
| 0       | START    | Start.<br>The PVP_UDS_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |
|         |          | 1      Enable (W1A)                                                                          |

## UDS Control

The PVP\_UDS\_CTL register selects operation modes for the up-down scaler (UDS) block, including filter tap selection.

**PVP\_UDS\_CTL: UDS Control - R/W**

Reset = 0x0000 0000

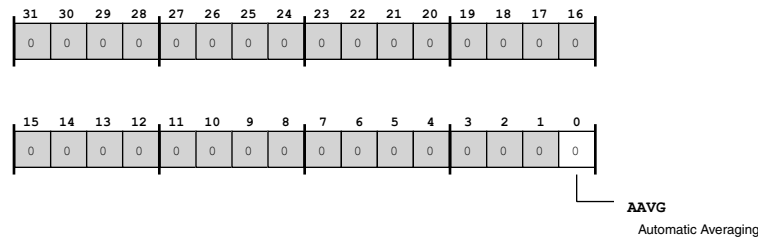


**Figure 30-95:** PVP\_UDS\_CTL Register Diagram

**Table 30-61:** PVP\_UDS\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                     |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------|
| 0       | AAVG     | Automatic Averaging.<br>The PVP_UDS_CTL.AAVG selects automatic or manual filter tap selection for the anti-aliasing filter. |
|         |          | 0      Manual Filter Tap Selection                                                                                          |
|         |          | 1      Auto Filter Tap Selection                                                                                            |

## UDS Output HCNT

The PVP\_UDS\_OHCNT selects the horizontal dimension for the output frame. Legal values for the target frame are: 16, 32, 64, 80, 96, 112, and 128.

PVP\_UDS\_OHCNT: UDS Output HCNT - R/W

Reset = 0x0000 0000

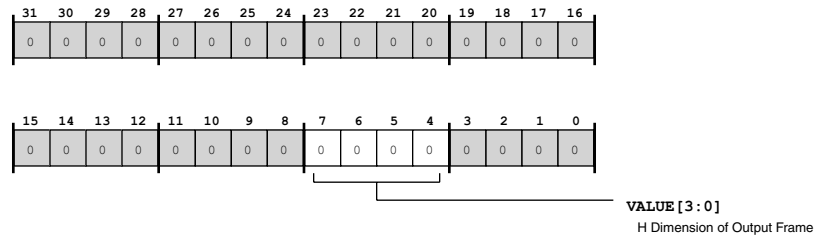


Figure 30-96: PVP\_UDS\_OHCNT Register Diagram

Table 30-62: PVP\_UDS\_OHCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration      |
|---------|----------|------------------------------|
| 7:4     | VALUE    | H Dimension of Output Frame. |

## UDS Output VCNT

The PVP\_UDS\_OVCNT selects the vertical dimension for the output frame. Legal values for the target frame are: 16, 32, 64, 80, 96, 112, and 128.

PVP\_UDS\_OVCNT: UDS Output VCNT - R/W

Reset = 0x0000 0000

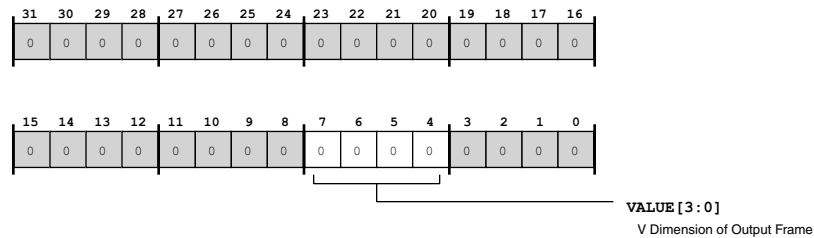


Figure 30-97: PVP\_UDS\_OVCNT Register Diagram



**Table 30-63:** PVP\_UDS\_OVCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration      |
|---------|----------|------------------------------|
| 7:4     | VALUE    | V Dimension of Output Frame. |

UDS HAVG

The PVP\_UDS\_HAVG selects the UDS H taps. Legal values for PVP\_UDS\_HAVG are between 1 and 128.

PVP\_UDS\_HAVG: UDS HAVG - R/W

Reset = 0x0000 0000

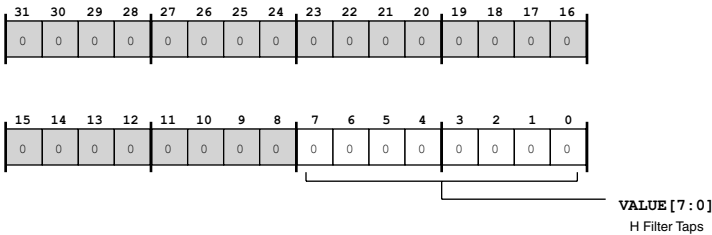


**Figure 30-98:** PVP\_UDS\_HAVG Register Diagram

**Table 30-64:** PVP\_UDS\_HAVG Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:0     | VALUE    | H Filter Taps.          |

UDS VAVG

The PVP\_UDS\_VAVG selects the UDS V taps. Legal values for PVP\_UDS\_VAVG are between 1 and 64.

PVP\_UDS\_VAVG: UDS VAVG - R/W

Reset = 0x0000 0000

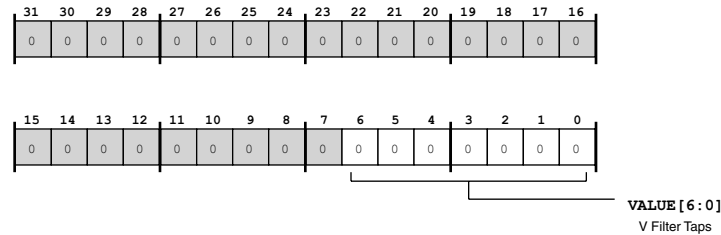


Figure 30-99: PVP\_UDS\_VAVG Register Diagram

Table 30-65: PVP\_UDS\_VAVG Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 6:0     | VALUE    | V Filter Taps.          |

IPF0 (Camera Pipe) Configuration

The input data formatter (IPF) blocks house FIFOs that buffer data for a sustained peak bandwidth output. The PVP\_IPF0\_CFG register enables the block's FIFO and configures blocks' connections within thePVP.

PVP\_IPF0\_CFG: IPF0 (Camera Pipe) Configuration - R/W

Reset = 0x0000 0000

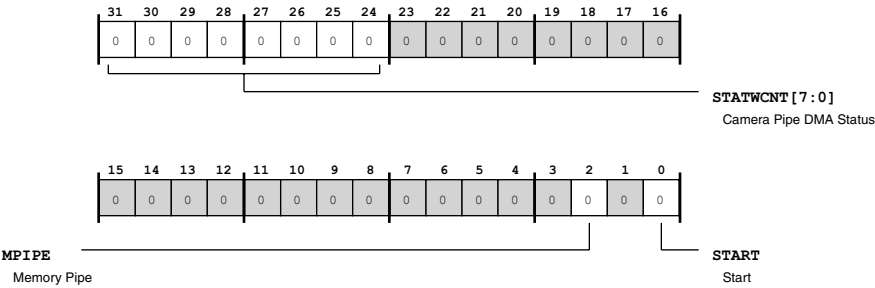


Figure 30-100: PVP\_IPF0\_CFG Register Diagram

**Table 30-66:** PVP\_IPF0\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:24   | STATWCNT | Camera Pipe DMA Status.<br>The PVP_IPF0_CFG.STATWCNT selects whether or not tag status is output on camera pipe status DMA channel. Note that all other values are reserved. |
|         |          | 0                      No Status Output                                                                                                                                      |
|         |          | 1                      Tag Status Output                                                                                                                                     |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_IPF0_CFG.MPIPE bit always reads as 0 (camera pipe).                                                                                                  |
| 0       | START    | Start.<br>The PVP_IPF0_CFG.START enables the PVP. This bit reads 0 after the pipe is drained.                                                                                |
|         |          | 1                      Enable (W1A)                                                                                                                                          |

## IPFn (Camera/Memory Pipe) Pipe Control

The PVP\_IPFn\_PIPECTL register controls the IPFn block's pipeline features, including status transmission and drain operation.

**PVP\_IPFn\_PIPECTL: IPFn (Camera/Memory Pipe) Pipe Control - R/W**

Reset = 0x0000 0000

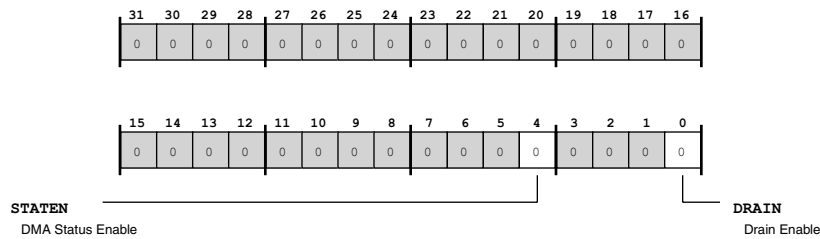


**Figure 30-101:** PVP\_IPFn\_PIPECTL Register Diagram

Table 30-67: PVP\_IPFn\_PIPECTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                         |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | STATEN   | DMA Status Enable.<br>The PVP_IPFn_PIPECTL.STATEN enables the IPF to send status DMA when PVP_IPFn_FCNT expires or every frame where PVP_IPFn_FCNT equals zero. |
|         |          | 0                      Disable                                                                                                                                  |
|         |          | 1                      Enable                                                                                                                                   |
| 0       | DRAIN    | Drain Enable.<br>The PVP_IPFn_PIPECTL.DRAIN enables the IPF to flush frames automatically in conjunction with PVP_IPFn_FCNT.                                    |
|         |          | 0                      Disable                                                                                                                                  |
|         |          | 1                      Enable                                                                                                                                   |

IPFn (Camera/Memory Pipe) Control

The PVP\_IPFn\_CTL register controls the IPFn block's pipeline features, including color format, format conversion, and output port selection.

PVP\_IPFn\_CTL: IPFn (Camera/Memory Pipe) Control - R/W

Reset = 0x0000 0000

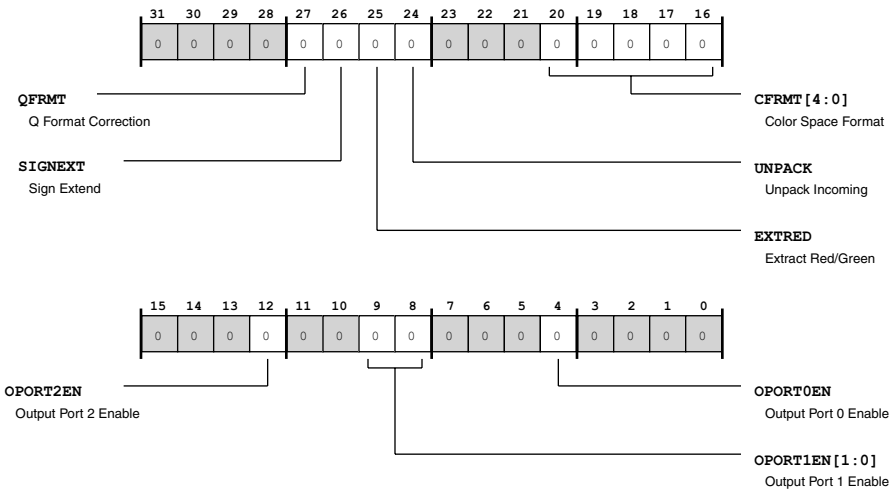


Figure 30-102: PVP\_IPFn\_CTL Register Diagram

**Table 30-68:** PVP\_IPFn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------|
| 27      | QFRMT    | Q Format Correction.<br>The PVP_IPFn_CTL.QFRMT enables Q format correction, converting data to unsigned by 1 left shift.                     |
|         |          | 0      Disable Q Format Correction                                                                                                           |
|         |          | 1      Enable Q Format Correction                                                                                                            |
| 26      | SIGNEXT  | Sign Extend.<br>The PVP_IPFn_CTL.SIGNEXT selects sign or zero extension for the extracted 5-, 6-, 8-, 16-, or 24-bit value to 32-bit value.  |
|         |          | 0      Zero Extend                                                                                                                           |
|         |          | 1      Sign Extend                                                                                                                           |
| 25      | EXTRED   | Extract Red/Green.<br>The PVP_IPFn_CTL.EXTRED selects red or green extraction from RGB format. This selection does not apply for YUV format. |
|         |          | 0      Extract Green                                                                                                                         |
|         |          | 1      Extract Red                                                                                                                           |
| 24      | UNPACK   | Unpack Incoming.<br>The PVP_IPFn_CTL.UNPACK enables unpacking for incoming data.                                                             |
|         |          | 0      No Unpacking                                                                                                                          |
|         |          | 1      Unpack Data                                                                                                                           |

**Table 30-68:** PVP\_IPFn\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------|
| 20:16   | CFRMT    | Color Space Format.<br>The PVP_IPFn_CTL.CFRMT field selects the Color Space Format. Note that all values other than those listed are reserved. |
|         |          | 0 RGB 8-Bit                                                                                                                                    |
|         |          | 1 RGB 888                                                                                                                                      |
|         |          | 2 RGB 565                                                                                                                                      |
|         |          | 3 RGB 666                                                                                                                                      |
|         |          | 4 RGB 16-Bit                                                                                                                                   |
|         |          | 5 RGB Bayer Format Type-1                                                                                                                      |
|         |          | 6 RGB Bayer Format Type-2                                                                                                                      |
|         |          | 16 YUV 4:2:2 8-Bit Type 1                                                                                                                      |
|         |          | 17 YUV 4:2:2 8-Bit Type 2<br>Includes with Color Split; this format is only supported in packed mode.                                          |
|         |          | 18 YUV 4:2:2 8-Bit Type 3<br>Includes Color Split and Sub-Split; this format is only supported in packed mode.                                 |
|         |          | 19 YUV 4:2:2 8-Bit Pair 16-Bit                                                                                                                 |
|         |          | 20 YUV 4:2:2 16-Bit Type 1                                                                                                                     |
|         |          | 21 YUV 4:2:2 16-Bit Type 2<br>Includes Color Split; this format is only supported in packed mode.                                              |
|         |          | 22 YUV 4:2:2 16-Bit Type 3<br>Includes Color Split and Sub-Split; this format is only supported in packed mode.                                |
|         |          | 24 Y Alone 8-Bit                                                                                                                               |
|         |          | 25 Y Alone 16-Bit                                                                                                                              |
|         |          | 26 Y Alone 24-Bit                                                                                                                              |

**Table 30-68:** PVP\_IPFn\_CTL Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                               |
|---------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | OPOINT2EN | Output Port 2 Enable.<br>The PVP_IPFn_CTL.OPOINT2EN enables OPOINT2 to send even pixels of full resolution. The send occurs after color extraction.                                                                                                                                                                   |
|         |           | 0                      Disable OPOINT2                                                                                                                                                                                                                                                                                |
|         |           | 1                      Enable OPOINT2 (full resolution)                                                                                                                                                                                                                                                               |
| 9:8     | OPOINT1EN | Output Port 1 Enable.<br>The PVP_IPFn_CTL.OPOINT1EN enables OPOINT1 to send all pixels of windowed resolution or to 2 to send odd pixels of full resolution. The send occurs after color extraction. Note that the windowed resolution selection is valid only for camera pipe IPF and is invalid for memory pipe IPF |
|         |           | 0                      Disable OPOINT1                                                                                                                                                                                                                                                                                |
|         |           | 1                      Enable OPOINT1 (full resolution)                                                                                                                                                                                                                                                               |
|         |           | 2                      Enable OPOINT1 (windowed resolution)                                                                                                                                                                                                                                                           |
| 4       | OPOINT0EN | Output Port 0 Enable.<br>The PVP_IPFn_CTL.OPOINT0EN enables OPOINT0.                                                                                                                                                                                                                                                  |
|         |           | 0                      Disable OPOINT0                                                                                                                                                                                                                                                                                |
|         |           | 1                      Enable OPOINT0                                                                                                                                                                                                                                                                                 |

## IPFn (Camera/Memory Pipe) TAG Value

The PVP\_IPFn\_TAG holds a unique, user-programmable tag value that indicates to which frame the values were applied. If there are situations where the software does not need information about whether or not a new control word was applied for a new frame, the tag need not be programmed.

PVP\_IPFn\_TAG: IPFn (Camera/Memory Pipe) TAG Value - R/W

Reset = 0x0000 0000

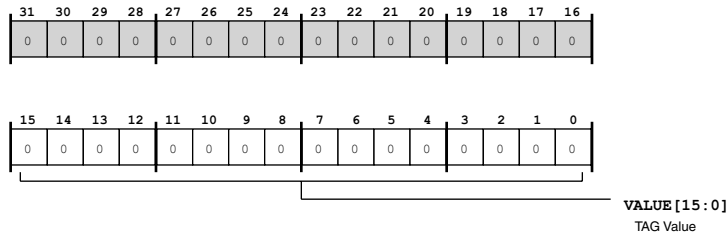


Figure 30-103: PVP\_IPFn\_TAG Register Diagram

Table 30-69: PVP\_IPFn\_TAG Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | TAG Value.              |

IPFn (Camera/Memory Pipe) Frame Count

The PVP\_IPFn\_FCNT whether a given block control structure list (BCL) applies to a set of frames or applies for every new frame.

PVP\_IPFn\_FCNT: IPFn (Camera/Memory Pipe) Frame Count - R/W

Reset = 0x0000 0000

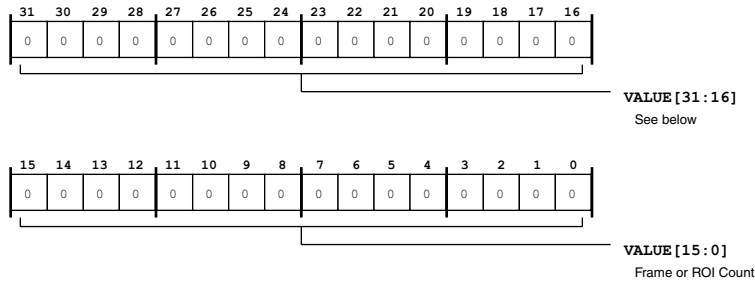


Figure 30-104: PVP\_IPFn\_FCNT Register Diagram

Table 30-70: PVP\_IPFn\_FCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Frame or ROI Count.     |



IPFn (Camera/Memory Pipe) Horizontal Count

The PVP\_IPFn\_HCNT contains the horizontal count (in pixels) for the region of interest (ROI) starting at the PVP\_IPF0\_HPOS position.

PVP\_IPFn\_HCNT: IPFn (Camera/Memory Pipe) Horizontal Count - R/W

Reset = 0x0000 0000

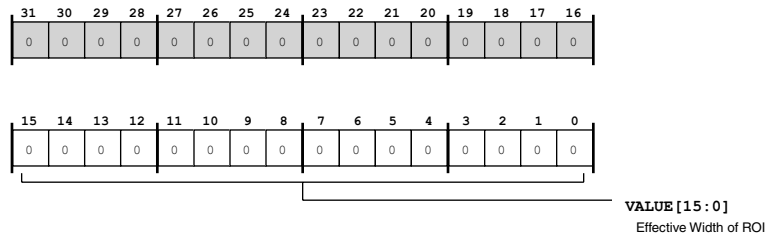


Figure 30-105: PVP\_IPFn\_HCNT Register Diagram

Table 30-71: PVP\_IPFn\_HCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Effective Width of ROI. |

IPFn (Camera/Memory Pipe) Vertical Count

The PVP\_IPFn\_VCNT contains the vertical count (in pixels) for the region of interest (ROI) starting at the PVP\_IPF0\_VPOS position.

PVP\_IPFn\_VCNT: IPFn (Camera/Memory Pipe) Vertical Count - R/W

Reset = 0x0000 0000

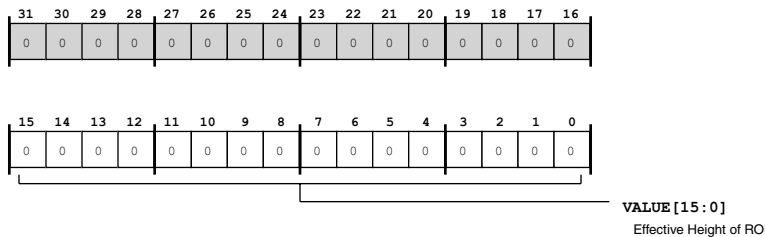


Figure 30-106: PVP\_IPFn\_VCNT Register Diagram

Table 30-72: PVP\_IPFn\_VCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 15:0    | VALUE    | Effective Height of ROI. |

IPF0 (Camera Pipe) Horizontal Position

The PVP\_IPF0\_HPOS contains the horizontal starting position (in pixels) for the region of interest (ROI).

PVP\_IPF0\_HPOS: IPF0 (Camera Pipe) Horizontal Position - R/W

Reset = 0x0000 0000

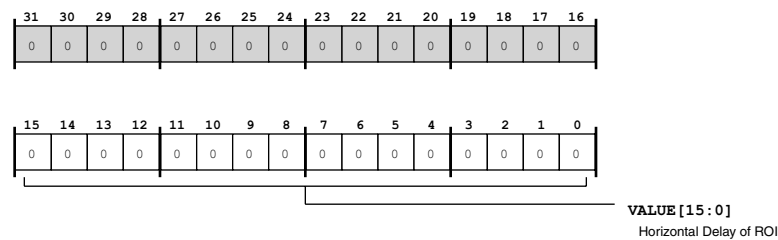


Figure 30-107: PVP\_IPF0\_HPOS Register Diagram

Table 30-73: PVP\_IPF0\_HPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 15:0    | VALUE    | Horizontal Delay of ROI. |

IPF0 (Camera Pipe) Vertical Position

The PVP\_IPF0\_VPOS contains the vertical starting position (in pixels) for the region of interest (ROI).

**PVP\_IPF0\_VPOS: IPF0 (Camera Pipe) Vertical Position - R/W**

Reset = 0x0000 0000

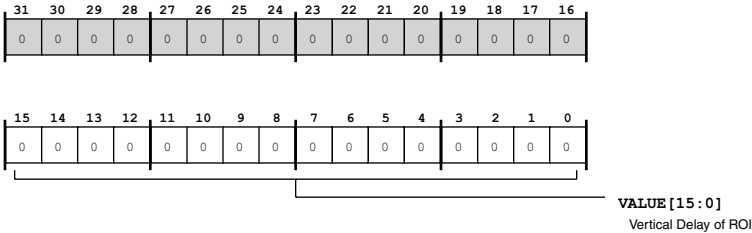


**Figure 30-108:** PVP\_IPF0\_VPOS Register Diagram

**Table 30-74:** PVP\_IPF0\_VPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Vertical Delay of ROI.  |

**IPF<sub>n</sub> (Camera/Memory Pipe) TAG Status**

After daisy chain load is completed, the value in PVP\_IPF<sub>n</sub>\_TAG is loaded to the PVP\_IPF<sub>n</sub>\_TAG\_STAT. This register is read-only register and may also be read through status read DMA. The PVP\_IPF<sub>n</sub>\_TAG\_STAT register is double buffered internally with the PVP\_IPF<sub>n</sub>\_TAG\_STAT corresponding to previous PVP\_IPF<sub>n</sub>\_FCNT entity is sent on status DMA.

**PVP\_IPF<sub>n</sub>\_TAG\_STAT: IPF<sub>n</sub> (Camera/Memory Pipe) TAG Status - R/NW**

Reset = 0x0000 0000

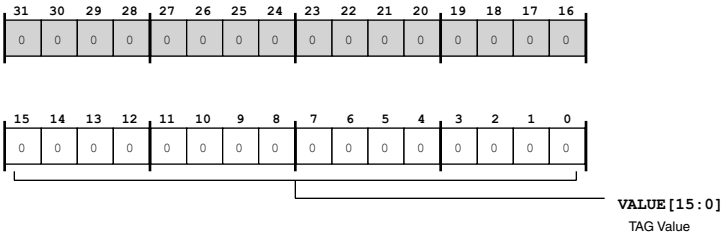


**Figure 30-109:** PVP\_IPF<sub>n</sub>\_TAG\_STAT Register Diagram

**Table 30-75:** PVP\_IPF<sub>n</sub>\_TAG\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | TAG Value.              |

IPF1 (Memory Pipe) Configuration

The PVP\_IPF1\_CFG register controls the IPF1 block's pipeline features, including status transmission and drain operation.

PVP\_IPF1\_CFG: IPF1 (Memory Pipe) Configuration - R/W

Reset = 0x0000 0004

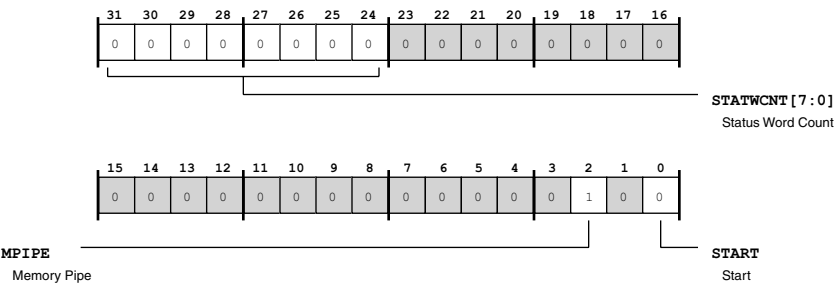


Figure 30-110: PVP\_IPF1\_CFG Register Diagram

Table 30-76: PVP\_IPF1\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                             |                   |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|
| 31:24   | STATWCNT | Status Word Count.<br>The PVP_IPF1_CFG.STATWCNT select the camera pipe DMA status information to be output on the memory pipe DMA channel. Note that all other values are reserved. |                   |
|         |          | 0                                                                                                                                                                                   | No Status Output  |
|         |          | 1                                                                                                                                                                                   | Tag Status Output |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_IPF1_CFG.MPIPE bit always reads as 1 (memory pipe).                                                                                                         |                   |
| 0       | START    | Start.<br>The PVP_IPF1_CFG.START enables the PVP. This bit reads 0 after the pipe is drained.                                                                                       |                   |
|         |          | 1                                                                                                                                                                                   | Enable (W1A)      |

CNVn Configuration

The PVP\_CNVn\_CFG register controls the convolution/down-scaler engine (CNV) block's pipeline features, including status transmission and drain operation.

PVP\_CNVn\_CFG: CNVn Configuration - R/W

Reset = 0x0000 0000

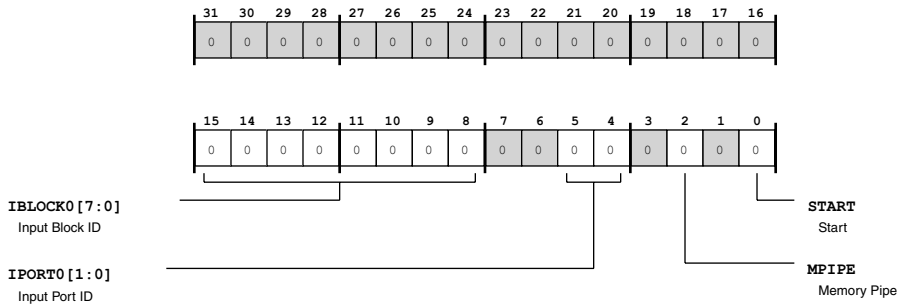


Figure 30-111: PVP\_CNVn\_CFG Register Diagram

Table 30-77: PVP\_CNVn\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_CNVn_CFG.IBLOCK0 and PVP_CNVn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). |
|         |          | 12 IPF0                                                                                                                                                                                                                   |
|         |          | 14 IPF1                                                                                                                                                                                                                   |
|         |          | 16 CNV0                                                                                                                                                                                                                   |
|         |          | 20 CNV1                                                                                                                                                                                                                   |
|         |          | 24 CNV2                                                                                                                                                                                                                   |
|         |          | 28 CNV3                                                                                                                                                                                                                   |

Table 30-77: PVP\_CNVn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_CNVn_CFG.IBLOCK0 and PVP_CNVn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_CNVn_CFG.MPIPE bit selects memory pipe or camera pipe.                                                                                                                                                                                                                                                         |
|         |          | 0      Camera Pipe                                                                                                                                                                                                                                                                                                                     |
|         |          | 1      Memory Pipe                                                                                                                                                                                                                                                                                                                     |
| 0       | START    | Start.<br>The PVP_CNVn_CFG.START enables the PVP. This bit reads 0 after the pipe is drained.                                                                                                                                                                                                                                          |
|         |          | 1      Enable (W1A)                                                                                                                                                                                                                                                                                                                    |

CNVn Control

The PVP\_CNVn\_CTL register controls the CNVn block's convolution features, including output saturation, output right shift, and zero/data fill selection.

PVP\_CNVn\_CTL: CNVn Control - R/W

Reset = 0x0000 0000

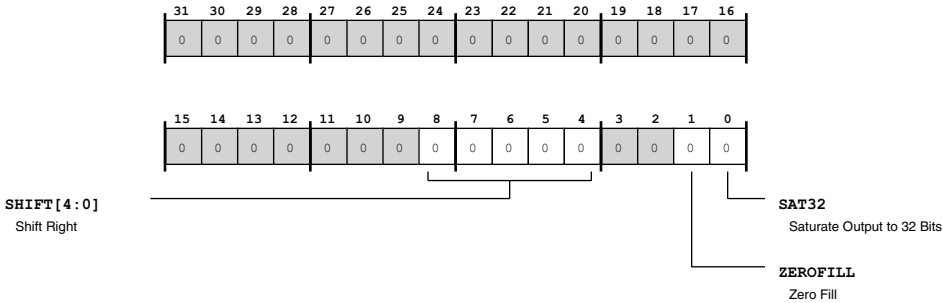


Figure 30-112: PVP\_CNVn\_CTL Register Diagram

**Table 30-78:** PVP\_CNVn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8:4     | SHIFT    | Shift Right.<br>The PVP_CNVn_CTL.SHIFT selects arithmetic right shift of output from 0 to 31 bits.                                                                                                                                                          |
| 1       | ZEROFILL | Zero Fill.<br>The PVP_CNVn_CTL.ZEROFILL select whether to fill edge data with zeroes or with duplicated pixels.                                                                                                                                             |
|         |          | 0 Duplicated Data Fill                                                                                                                                                                                                                                      |
|         |          | 1 Zero Fill                                                                                                                                                                                                                                                 |
| 0       | SAT32    | Saturate Output to 32 Bits.<br>The PVP_CNVn_CTL.SAT32 selects whether to saturate output to highest 32-bit value or highest 16-bit value. For 16-bit saturate of output, the value is driven on the lower 16 bits, and the upper 16 bits are sign-extended. |
|         |          | 0 16-Bit Saturate of Output                                                                                                                                                                                                                                 |
|         |          | 1 32-Bit Saturate of Output                                                                                                                                                                                                                                 |

## CNVn Coefficients 0,0 and 0,1

The PVP\_CNVn\_C00C01 provides coefficients 0,0 and 0,1 for the convolution matrix.

**PVP\_CNVn\_C00C01: CNVn Coefficients 0,0 and 0,1 - R/W**

Reset = 0x0000 0000

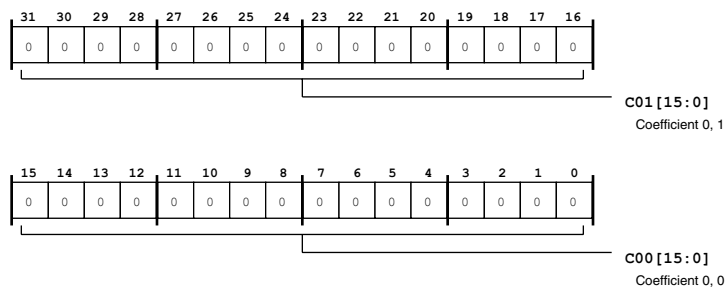


**Figure 30-113:** PVP\_CNVn\_C00C01 Register Diagram

Table 30-79: PVP\_CNVn\_C00C01 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C01      | Coefficient 0, 1.       |
| 15:0    | C00      | Coefficient 0, 0.       |

CNVn Coefficients 0,2 and 0,3

The PVP\_CNVn\_C02C03 provides coefficients 0,2 and 0,3 for the convolution matrix.

PVP\_CNVn\_C02C03: CNVn Coefficients 0,2 and 0,3 - R/W

Reset = 0x0000 0000

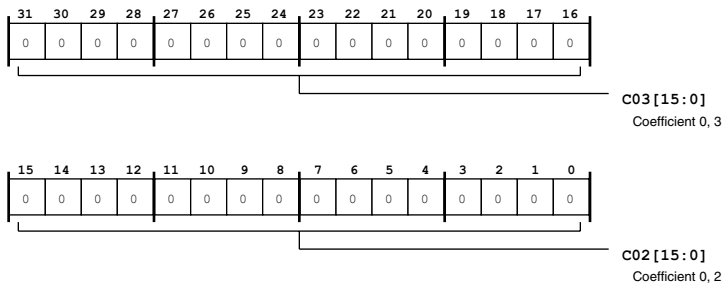


Figure 30-114: PVP\_CNVn\_C02C03 Register Diagram

Table 30-80: PVP\_CNVn\_C02C03 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C03      | Coefficient 0, 3.       |
| 15:0    | C02      | Coefficient 0, 2.       |

CNVn Coefficient 0,4

The PVP\_CNVn\_C04 provides coefficients 0,4 for the convolution matrix.



**PVP\_CNVn\_C04: CNVn Coefficient 0,4 - R/W**

Reset = 0x0000 0000

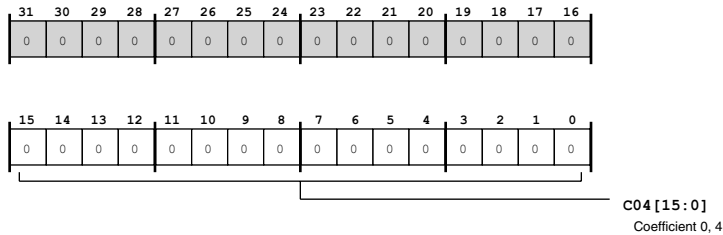


**Figure 30-115:** PVP\_CNVn\_C04 Register Diagram

**Table 30-81:** PVP\_CNVn\_C04 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C04      | Coefficient 0, 4.       |

## CNVn Coefficients 1,0 and 1,1

The PVP\_CNVn\_C10C11 provides coefficients 1,0 and 1,1 for the convolution matrix.

**PVP\_CNVn\_C10C11: CNVn Coefficients 1,0 and 1,1 - R/W**

Reset = 0x0000 0000

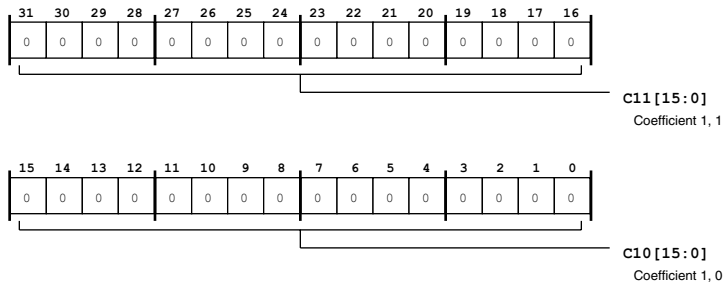


**Figure 30-116:** PVP\_CNVn\_C10C11 Register Diagram

**Table 30-82:** PVP\_CNVn\_C10C11 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C11      | Coefficient 1, 1.       |
| 15:0    | C10      | Coefficient 1, 0.       |

CNVn Coefficients 1,2 and 1,3

The PVP\_CNVn\_C12C13 provides coefficients 1,2 and 1,3 for the convolution matrix.

PVP\_CNVn\_C12C13: CNVn Coefficients 1,2 and 1,3 - R/W

Reset = 0x0000 0000

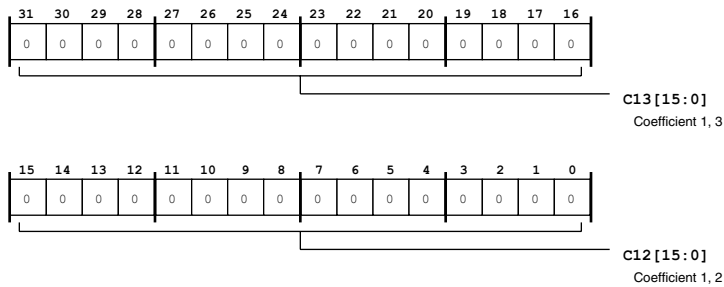


Figure 30-117: PVP\_CNVn\_C12C13 Register Diagram

Table 30-83: PVP\_CNVn\_C12C13 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C13      | Coefficient 1, 3.       |
| 15:0    | C12      | Coefficient 1, 2.       |

CNVn Coefficient 1,4

The PVP\_CNVn\_C14 provides coefficients 1,4 for the convolution matrix.

PVP\_CNVn\_C14: CNVn Coefficient 1,4 - R/W

Reset = 0x0000 0000

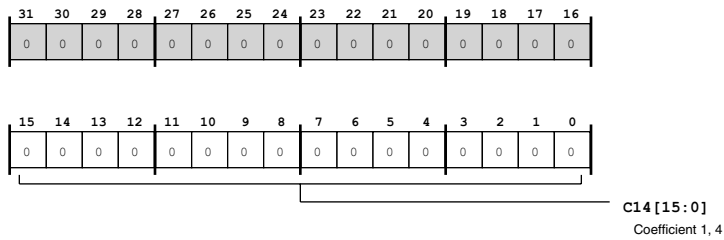


Figure 30-118: PVP\_CNVn\_C14 Register Diagram

**Table 30-84:** PVP\_CNVn\_C14 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C14      | Coefficient 1, 4.       |

CNVn Coefficients 2,0 and 2,1

The PVP\_CNVn\_C20C21 provides coefficients 2,0 and 2,1 for the convolution matrix.

PVP\_CNVn\_C20C21: CNVn Coefficients 2,0 and 2,1 - R/W

Reset = 0x0000 0000

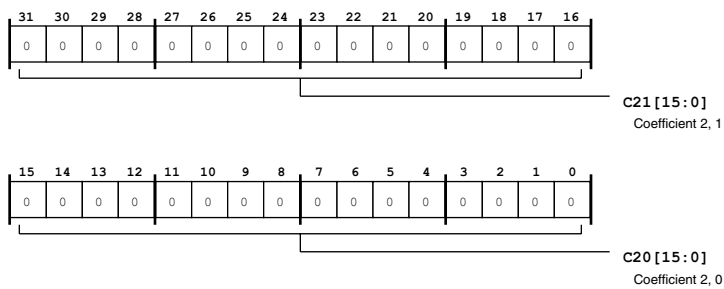


**Figure 30-119:** PVP\_CNVn\_C20C21 Register Diagram

**Table 30-85:** PVP\_CNVn\_C20C21 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C21      | Coefficient 2, 1.       |
| 15:0    | C20      | Coefficient 2, 0.       |

CNVn Coefficients 2,2 and 2,3

The PVP\_CNVn\_C22C23 provides coefficients 2,2 and 2,3 for the convolution matrix.

PVP\_CNVn\_C22C23: CNVn Coefficients 2,2 and 2,3 - R/W

Reset = 0x0000 0000

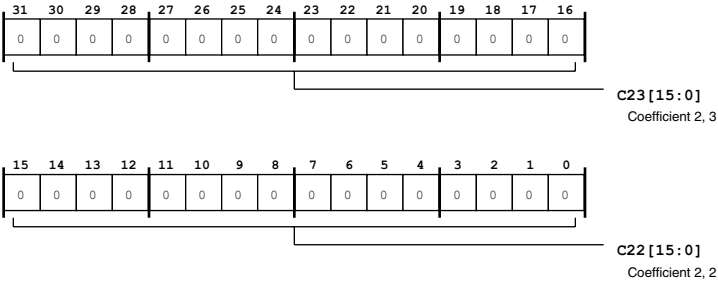


Figure 30-120: PVP\_CNVn\_C22C23 Register Diagram

Table 30-86: PVP\_CNVn\_C22C23 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C23      | Coefficient 2, 3.       |
| 15:0    | C22      | Coefficient 2, 2.       |

CNVn Coefficient 2,4

The PVP\_CNVn\_C24 provides coefficients 2,4 for the convolution matrix.

PVP\_CNVn\_C24: CNVn Coefficient 2,4 - R/W

Reset = 0x0000 0000

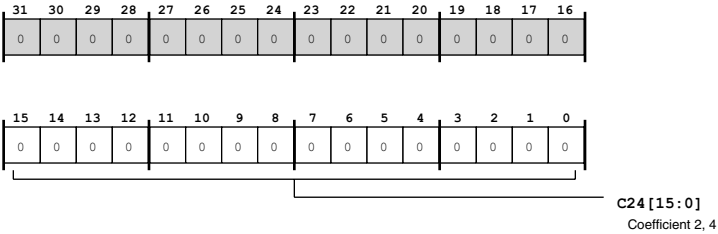


Figure 30-121: PVP\_CNVn\_C24 Register Diagram

Table 30-87: PVP\_CNVn\_C24 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C24      | Coefficient 2, 4.       |

## CNVn Coefficients 3,0 and 3,1

The PVP\_CNVn\_C30C31 provides coefficients 3,0 and 3,1 for the convolution matrix.

**PVP\_CNVn\_C30C31: CNVn Coefficients 3,0 and 3,1 - R/W**

Reset = 0x0000 0000

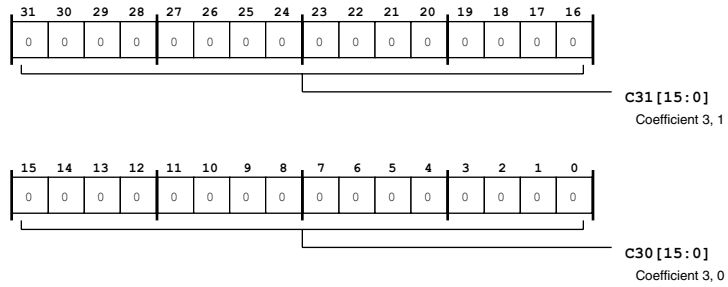


**Figure 30-122:** PVP\_CNVn\_C30C31 Register Diagram

**Table 30-88:** PVP\_CNVn\_C30C31 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C31      | Coefficient 3, 1.       |
| 15:0    | C30      | Coefficient 3, 0.       |

## CNVn Coefficients 3,2 and 3,3

The PVP\_CNVn\_C32C33 provides coefficients 3,2 and 3,3 for the convolution matrix.

**PVP\_CNVn\_C32C33: CNVn Coefficients 3,2 and 3,3 - R/W**

Reset = 0x0000 0000

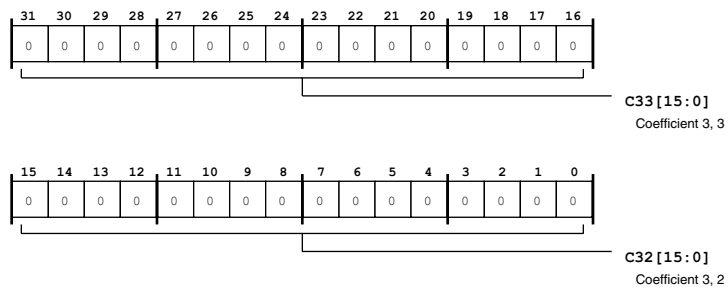


**Figure 30-123:** PVP\_CNVn\_C32C33 Register Diagram

Table 30-89: PVP\_CNVn\_C32C33 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C33      | Coefficient 3, 3.       |
| 15:0    | C32      | Coefficient 3, 2.       |

CNVn Coefficient 3,4

The PVP\_CNVn\_C34 provides coefficients 3,4 for the convolution matrix.

PVP\_CNVn\_C34: CNVn Coefficient 3,4 - R/W

Reset = 0x0000 0000

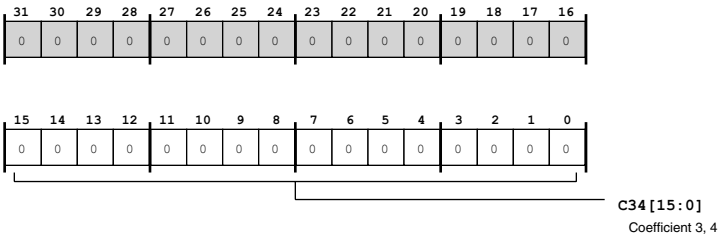


Figure 30-124: PVP\_CNVn\_C34 Register Diagram

Table 30-90: PVP\_CNVn\_C34 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C34      | Coefficient 3, 4.       |

CNVn Coefficients 4,0 and 4,1

The PVP\_CNVn\_C40C41 provides coefficients 4,0 and 4,1 for the convolution matrix.

**PVP\_CNVn\_C40C41: CNVn Coefficients 4,0 and 4,1 - R/W**

Reset = 0x0000 0000

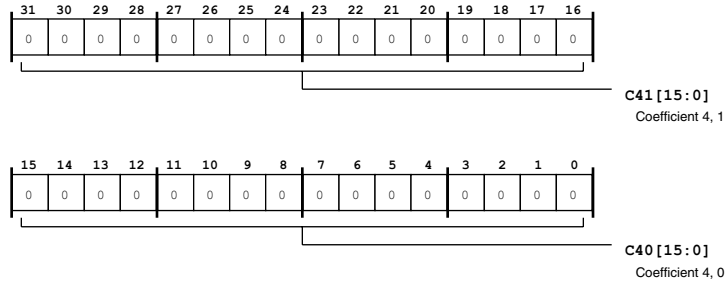


**Figure 30-125:** PVP\_CNVn\_C40C41 Register Diagram

**Table 30-91:** PVP\_CNVn\_C40C41 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C41      | Coefficient 4, 1.       |
| 15:0    | C40      | Coefficient 4, 0.       |

## CNVn Coefficients 4,2 and 4,3

The PVP\_CNVn\_C42C43 provides coefficients 4,2 and 4,3 for the convolution matrix.

**PVP\_CNVn\_C42C43: CNVn Coefficients 4,2 and 4,3 - R/W**

Reset = 0x0000 0000

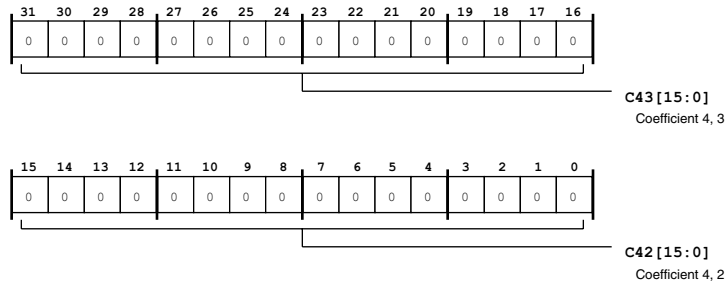


**Figure 30-126:** PVP\_CNVn\_C42C43 Register Diagram

**Table 30-92:** PVP\_CNVn\_C42C43 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | C43      | Coefficient 4, 3.       |
| 15:0    | C42      | Coefficient 4, 2.       |

Table 30-92: PVP\_CNVn\_C42C43 Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C42      | Coefficient 4, 2.       |

CNVn Coefficient 4,4

The PVP\_CNVn\_C44 provides coefficients 4,4 for the convolution matrix.

PVP\_CNVn\_C44: CNVn Coefficient 4,4 - R/W

Reset = 0x0000 0000

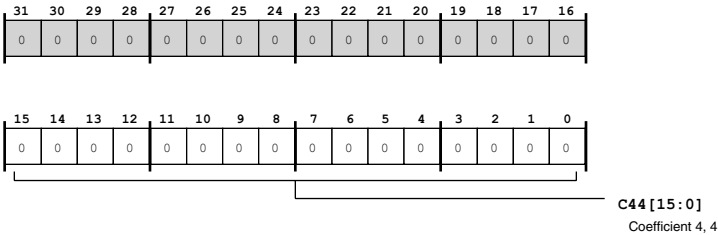


Figure 30-127: PVP\_CNVn\_C44 Register Diagram

Table 30-93: PVP\_CNVn\_C44 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | C44      | Coefficient 4, 4.       |

CNVn Scaling Factor

The PVP\_CNVn\_SCALE holds the vertical and horizontal scaling factors for down scaling incoming frames with decimation ratios of up to 256 (for example, 1, 2, 4, 8, 16, 32, 64, 128, or 256) in either direction.



**PVP\_CNvN\_SCALE: CNvN Scaling Factor - R/W**

Reset = 0x0000 0000

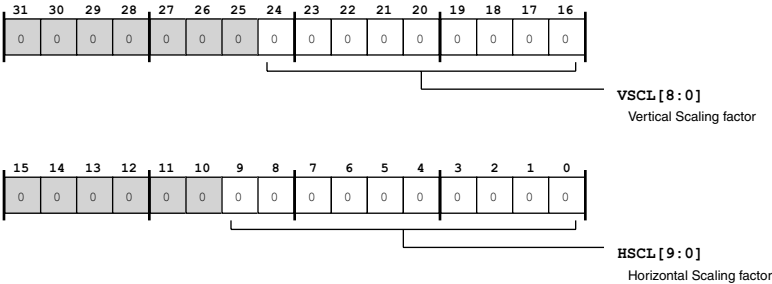


**Figure 30-128:** PVP\_CNvN\_SCALE Register Diagram

**Table 30-94:** PVP\_CNvN\_SCALE Register Fields

| Bit No. | Bit Name | Description/Enumeration    |
|---------|----------|----------------------------|
| 24:16   | VSCL     | Vertical Scaling factor.   |
| 9:0     | HSCL     | Horizontal Scaling factor. |

**THCn Configuration**

The PVP\_THCn\_CFG register controls the threshold/histogram and compression engine (THC) block's pipeline features, including status transmission, port selection, and block enable.

**PVP\_THCn\_CFG: THCn Configuration - R/W**

Reset = 0x0000 0000

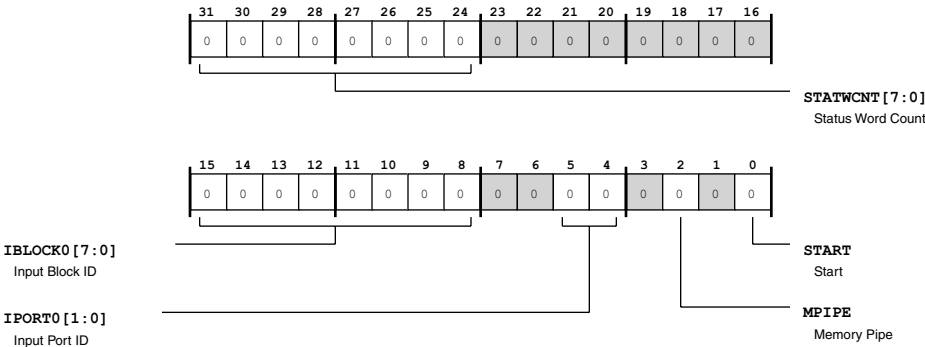


**Figure 30-129:** PVP\_THCn\_CFG Register Diagram

**Table 30-95:** PVP\_THCn\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:24   | STATWCNT | Status Word Count.<br>The PVP_THCn_CFG.STATWCNT select the camera pipe status information to be output on the memory pipe DMA channel. Note that all other values are reserved.                                                                                                                                                        |
| 15:8    | IBLOCK0  | Input Block ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_THCn_CFG.IBLOCK0 and PVP_THCn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 5 PEC                                                                                                                                                                                                                                                                                                                                  |
|         |          | 8 ACU                                                                                                                                                                                                                                                                                                                                  |
|         |          | 12 IPF0                                                                                                                                                                                                                                                                                                                                |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
|         |          | 48 PMA                                                                                                                                                                                                                                                                                                                                 |
| 5:4     | IPORT0   | Input Port ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_THCn_CFG.IBLOCK0 and PVP_THCn_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |

**Table 30-95:** PVP\_THCn\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------|
| 2       | MPIPE    | Memory Pipe.<br>The PVP_THCn_CFG.MPIPE bit selects memory pipe or camera pipe.                |
|         |          | 0 Camera Pipe                                                                                 |
|         |          | 1 Memory Pipe                                                                                 |
| 0       | START    | Start.<br>The PVP_THCn_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |
|         |          | 1 Enable (W1A)                                                                                |

THCn Control

The PVP\_THCn\_CTL register controls the THCn block's threshold/histogram features, including histogram window and counters, output format, and threshold mode selection.

PVP\_THCn\_CTL: THCn Control - R/W

Reset = 0x0000 0000

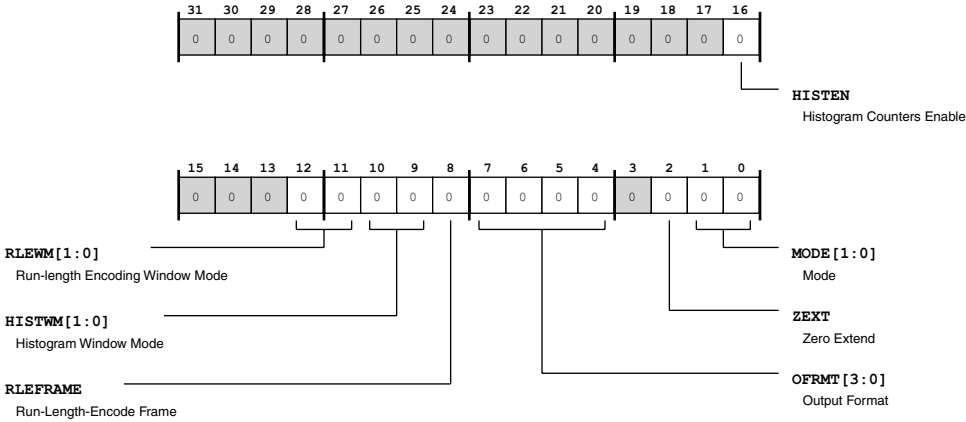


**Figure 30-130:** PVP\_THCn\_CTL Register Diagram

**Table 30-96:** PVP\_THCn\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | HISTEN   | Histogram Counters Enable.<br>The PVP_THCn_CTL.HISTEN enables histogram counters, enabling the histogram operation.                                                                                                                                                                                                                                                                                                           |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 12:11   | RLEWM    | Run-length Encoding Window Mode.<br>The PVP_THCn_CTL.RLEWM selects the window mode for the compression block, choosing either compression for pixels inside the window or compression for pixels in the entire frame. Note that values other than shown are invalid.                                                                                                                                                          |
|         |          | 0      Frame Compression                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1      Window Compression                                                                                                                                                                                                                                                                                                                                                                                                     |
| 10:9    | HISTWM   | Histogram Window Mode.<br>The PVP_THCn_CTL.HISTWM selects window mode for the histogram block, choosing either histogram for pixels inside the window, for pixels outside the window, or for pixels in the entire frame.                                                                                                                                                                                                      |
|         |          | 0      Frame Histogram                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Inside-Window Histogram                                                                                                                                                                                                                                                                                                                                                                                                |
|         |          | 2      Outside-Window Histogram                                                                                                                                                                                                                                                                                                                                                                                               |
|         |          | 3      Reserved                                                                                                                                                                                                                                                                                                                                                                                                               |
| 8       | RLEFRAME | Run-Length-Encode Frame.<br>The PVP_THCn_CTL.RLEFRAME selects the scope for run-length compression as row or frame. When row is selected, the compression block generates a report at the end of a row and the run-length counter is reset. When frame is selected, compression works across rows in the frame. Note that, due to the nature of compression, a variable number of reports are generated per row or per frame. |
|         |          | 0      Row (Line) Compression                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1      Frame Compression                                                                                                                                                                                                                                                                                                                                                                                                      |

**Table 30-96:** PVP\_THCn\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                     |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | OFRMT    | <p>Output Format.</p> <p>The PVP_THCn_CTL.OFRMT selects the THC block's output format, which controls how the compression value and run-length are grouped into a single word. Note that values other than those shown are invalid. Also note that, depending on the PVP_THCn_CTL.OFRMT value, the appropriate number of MSBs are driven with zero value to complete the 32 bit output.</p> |
|         |          | 0 32-Bit Word ( No Compression )                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Reserved                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 2 4-Bit Index ( No Compression)                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 3 4-Bit Index / 4-Bit Run Length                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 4 4-Bit Index / 4-Bit angle ( No Compression)                                                                                                                                                                                                                                                                                                                                               |
|         |          | 5 3-Bit Index / 5-Bit Run Length                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 6 4-Bit Index / 12-Bit Run Length                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 7 3-Bit Index / 13-Bit Run Length                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 8 4-Bit Index / 21-Bit Run Length                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 9 16-Bit Word / 16-Bit Run Length                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 10 Disable Output/RLE<br>No data is driven on the output port, and the RLE block is switched off.                                                                                                                                                                                                                                                                                           |
| 2       | ZEXT     | <p>Zero Extend.</p> <p>The PVP_THCn_CTL.ZEXT directs the THC to zero extend the lower 16 bits of the input port to 32 bits for magnitude and to use the upper 16 bits as angle. If PVP_THCn_CTL.ZEXT is disabled, the THC uses the entire 32 bits of the input port for magnitude and treats the angle as all zeroes.</p>                                                                   |
|         |          | 0 No Zero Extension                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Zero Extend                                                                                                                                                                                                                                                                                                                                                                               |

Table 30-96: PVP\_THCn\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1:0     | MODE     | Mode.<br>The PVP_THCn_CTL.MODE selects the THC mode as clipping mode, quantization mode, or hysteresis mode. For more information about these modes, see the PVP architectural concepts section. |
|         |          | 0 Clipping/Saturation Mode                                                                                                                                                                       |
|         |          | 1 Quantization Mode                                                                                                                                                                              |
|         |          | 2 Hysteresis Mode                                                                                                                                                                                |
|         |          | 3 Reserved                                                                                                                                                                                       |

THCn Histogram Frame Count

The PVP\_THCn\_HFCNT holds the number of frames for which the histogram counters accumulate. The PVP\_THCn\_HFCNT\_STAT register counts frames up to the value set in PVP\_THCn\_HFCNT for histogram operation.

PVP\_THCn\_HFCNT: THCn Histogram Frame Count - R/W

Reset = 0x0000 0001

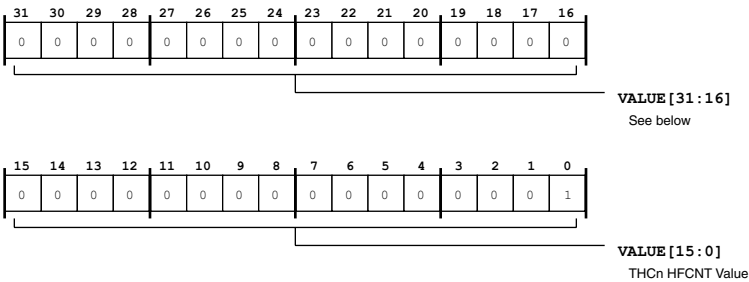


Figure 30-131: PVP\_THCn\_HFCNT Register Diagram

Table 30-97: PVP\_THCn\_HFCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | THCn HFCNT Value.       |

# THCn Max RLE Reports

This PVP\_THCn\_RMAXREP controls the maximum number of RLE reports per line or per frame.

PVP\_THCn\_RMAXREP: THCn Max RLE Reports - R/W

Reset = 0x0000 0020

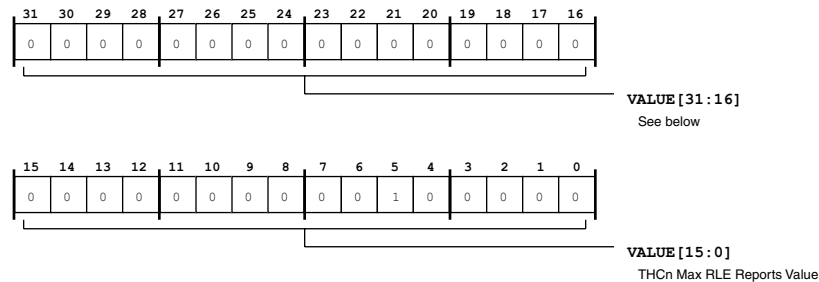


Figure 30-132: PVP\_THCn\_RMAXREP Register Diagram

Table 30-98: PVP\_THCn\_RMAXREP Register Fields

| Bit No. | Bit Name | Description/Enumeration     |
|---------|----------|-----------------------------|
| 31:0    | VALUE    | THCn Max RLE Reports Value. |

# THCn Min Clip Value

The PVP\_THCn\_CMINVAL holds the minimum clip value, which the THC uses in clipping/saturation mode. For more information about this mode and the registers involved in its operation (PVP\_THCn\_CMINVAL, PVP\_THCn\_CMINTH, PVP\_THCn\_CMAXTH,PVP\_THCn\_CMAXVAL), see the PVP functional description.

PVP\_THCn\_CMINVAL: THCn Min Clip Value - R/W

Reset = 0x8000 0000

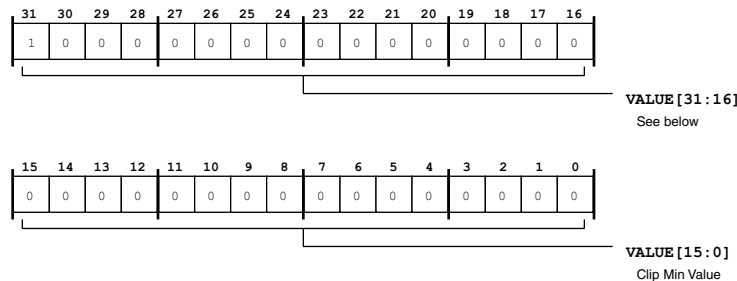


Figure 30-133: PVP\_THCn\_CMINVAL Register Diagram

Table 30-99: PVP\_THCn\_CMINVAL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Clip Min Value.         |

THCn Clip Min Threshold

The PVP\_THCn\_CMINTH holds the minimum clip value, which the THC uses in clipping/saturation mode. For more information about this mode and the registers involved in its operation (PVP\_THCn\_CMINVAL, PVP\_THCn\_CMINTH, PVP\_THCn\_CMAXTH,PVP\_THCn\_CMAXVAL), see the PVP functional description.

PVP\_THCn\_CMINTH: THCn Clip Min Threshold - R/W

Reset = 0x8000 0000

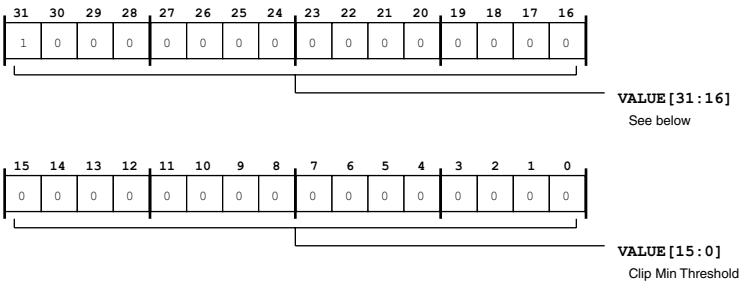


Figure 30-134: PVP\_THCn\_CMINTH Register Diagram

Table 30-100: PVP\_THCn\_CMINTH Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Clip Min Threshold.     |

THCn Clip Max Threshold

The PVP\_THCn\_CMAXTH holds the minimum clip value, which the THC uses in clipping/saturation mode. For more information about this mode and the registers involved in its operation (PVP\_THCn\_CMINVAL, PVP\_THCn\_CMINTH, PVP\_THCn\_CMAXTH,PVP\_THCn\_CMAXVAL), see the PVP functional description.



**PVP\_THCn\_CMAXTH: THCn Clip Max Threshold - R/W**

Reset = 0x7fff ffff

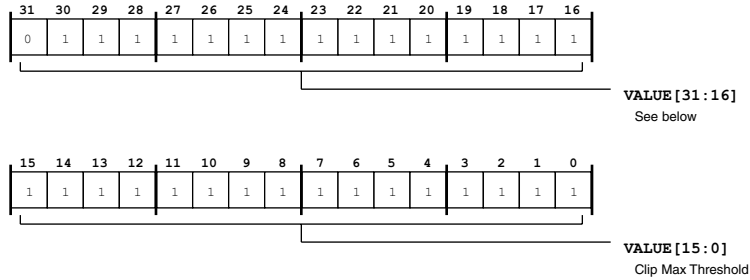


**Figure 30-135:** PVP\_THCn\_CMAXTH Register Diagram

**Table 30-101:** PVP\_THCn\_CMAXTH Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Clip Max Threshold.     |

**THCn Max Clip Value**

The PVP\_THCn\_CMAXVAL holds the minimum clip value, which the THC uses in clipping/saturation mode. For more information about this mode and the registers involved in its operation (PVP\_THCn\_CMINVAL, PVP\_THCn\_CMINTH, PVP\_THCn\_CMAXTH,PVP\_THCn\_CMAXVAL), see the PVP functional description.

**PVP\_THCn\_CMAXVAL: THCn Max Clip Value - R/W**

Reset = 0x7fff ffff

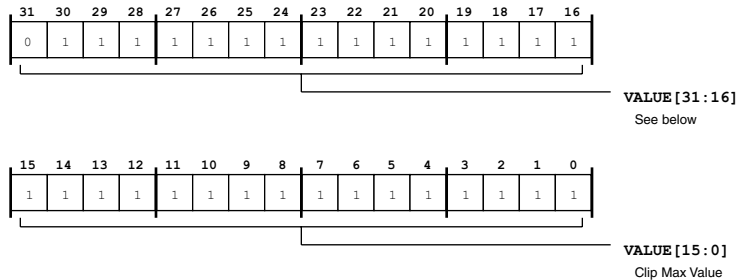


**Figure 30-136:** PVP\_THCn\_CMAXVAL Register Diagram

**Table 30-102:** PVP\_THCn\_CMAXVAL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Clip Max Value.         |

THCn Threshold Value 0

The PVP\_THCn\_TH0 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH0: THCn Threshold Value 0 - R/W

Reset = 0x7fff ffff

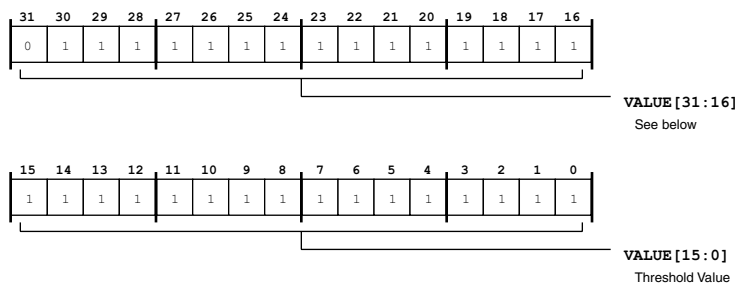


Figure 30-137: PVP\_THCn\_TH0 Register Diagram

Table 30-103: PVP\_THCn\_TH0 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 1

The PVP\_THCn\_TH1 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH1: THCn Threshold Value 1 - R/W

Reset = 0x7fff ffff

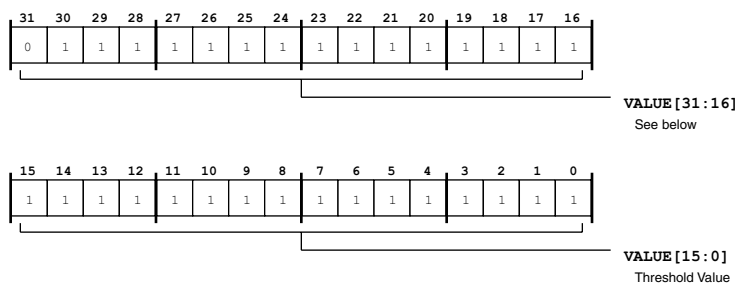


Figure 30-138: PVP\_THCn\_TH1 Register Diagram

**Table 30-104:** PVP\_THCn\_TH1 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

### THCn Threshold Value 2

The PVP\_THCn\_TH2 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

**PVP\_THCn\_TH2: THCn Threshold Value 2 - R/W**

Reset = 0x7fff ffff

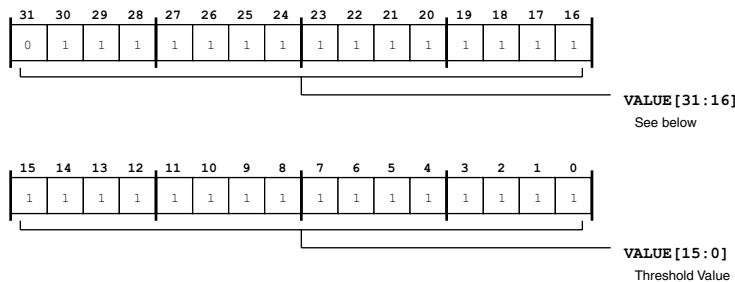


**Figure 30-139:** PVP\_THCn\_TH2 Register Diagram

**Table 30-105:** PVP\_THCn\_TH2 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

### THCn Threshold Value 3

The PVP\_THCn\_TH3 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH3: THCn Threshold Value 3 - R/W

Reset = 0x7fff ffff

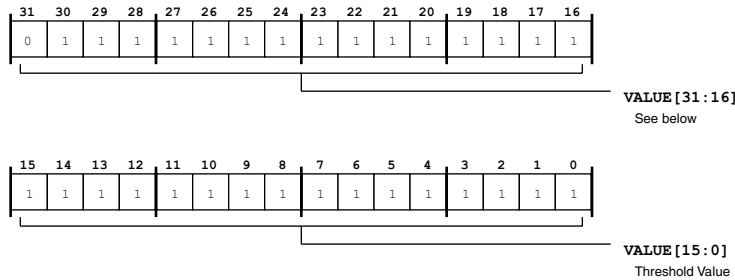


Figure 30-140: PVP\_THCn\_TH3 Register Diagram

Table 30-106: PVP\_THCn\_TH3 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 4

The PVP\_THCn\_TH4 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH4: THCn Threshold Value 4 - R/W

Reset = 0x7fff ffff

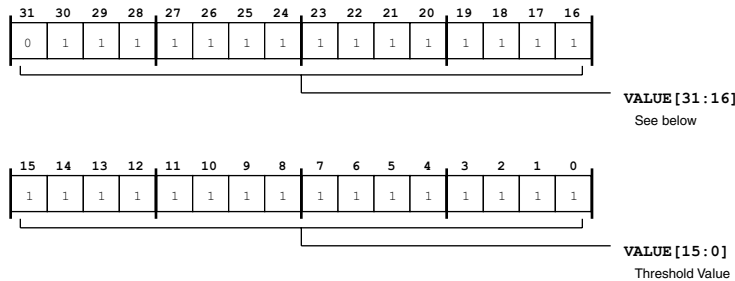


Figure 30-141: PVP\_THCn\_TH4 Register Diagram

Table 30-107: PVP\_THCn\_TH4 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 5

The PVP\_THCn\_TH5 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH5: THCn Threshold Value 5 - R/W

Reset = 0x7fff ffff

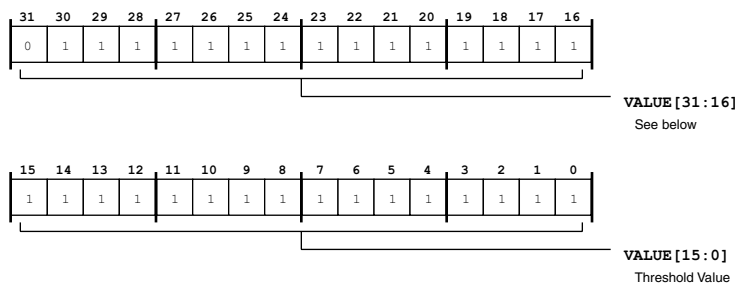


Figure 30-142: PVP\_THCn\_TH5 Register Diagram

Table 30-108: PVP\_THCn\_TH5 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 6

The PVP\_THCn\_TH6 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH6: THCn Threshold Value 6 - R/W

Reset = 0x7fff ffff

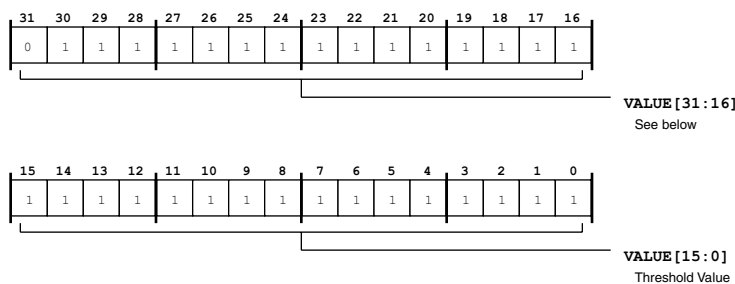


Figure 30-143: PVP\_THCn\_TH6 Register Diagram

Table 30-109: PVP\_THCn\_TH6 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 7

The PVP\_THCn\_TH7 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH7: THCn Threshold Value 7 - R/W

Reset = 0x7fff ffff

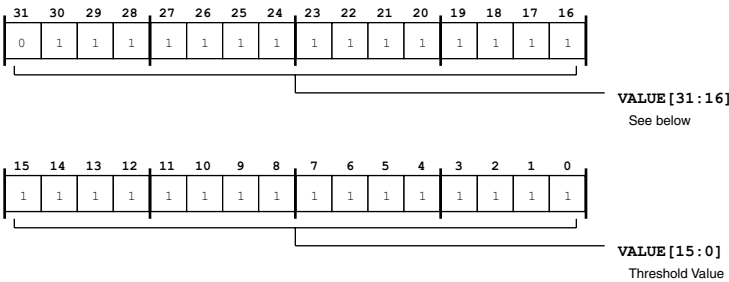


Figure 30-144: PVP\_THCn\_TH7 Register Diagram

Table 30-110: PVP\_THCn\_TH7 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 8

The PVP\_THCn\_TH8 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

**PVP\_THCn\_TH8: THCn Threshold Value 8 - R/W**

Reset = 0x7fff ffff

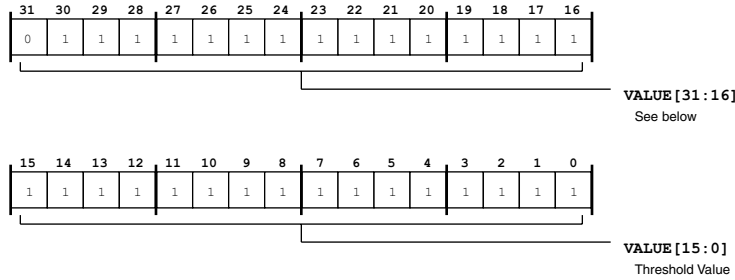


**Figure 30-145:** PVP\_THCn\_TH8 Register Diagram

**Table 30-111:** PVP\_THCn\_TH8 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

**THCn Threshold Value 9**

The PVP\_THCn\_TH9 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

**PVP\_THCn\_TH9: THCn Threshold Value 9 - R/W**

Reset = 0x7fff ffff

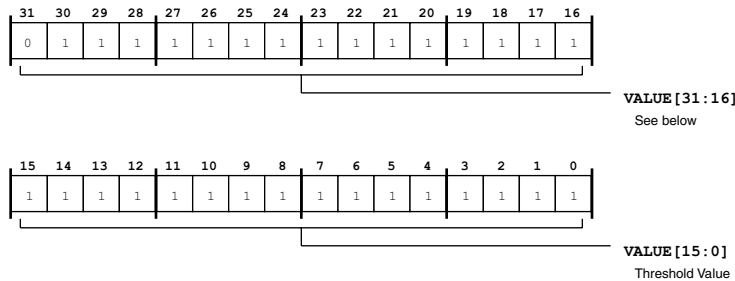


**Figure 30-146:** PVP\_THCn\_TH9 Register Diagram

**Table 30-112:** PVP\_THCn\_TH9 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

## THCn Threshold Value 10

The PVP\_THCn\_TH10 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH10: THCn Threshold Value 10 - R/W

Reset = 0x7fff ffff

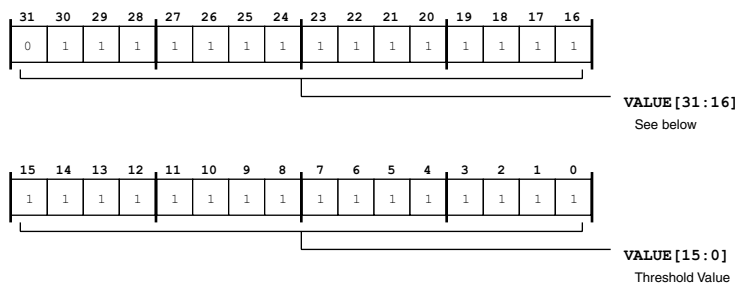


Figure 30-147: PVP\_THCn\_TH10 Register Diagram

Table 30-113: PVP\_THCn\_TH10 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

## THCn Threshold Value 11

The PVP\_THCn\_TH11 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH11: THCn Threshold Value 11 - R/W

Reset = 0x7fff ffff

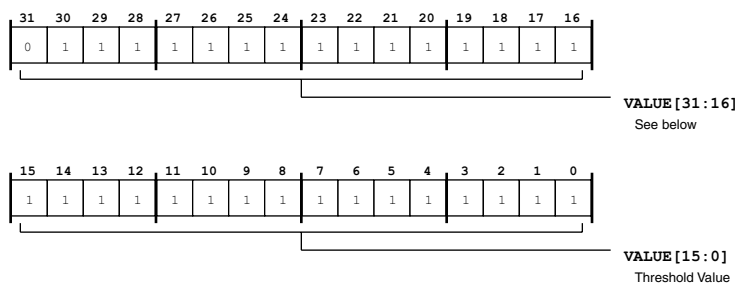


Figure 30-148: PVP\_THCn\_TH11 Register Diagram



**Table 30-114:** PVP\_THCn\_TH11 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

## THCn Threshold Value 12

The PVP\_THCn\_TH12 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

**PVP\_THCn\_TH12: THCn Threshold Value 12 - R/W**

Reset = 0x7fff ffff

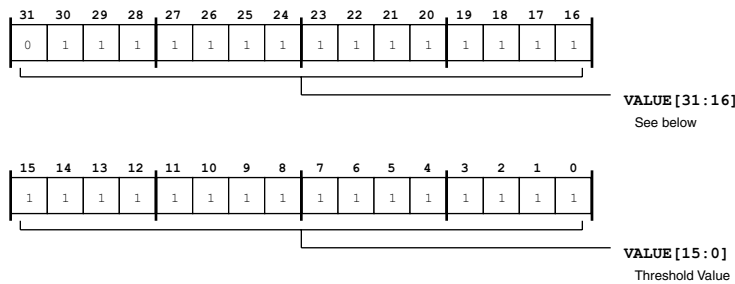


**Figure 30-149:** PVP\_THCn\_TH12 Register Diagram

**Table 30-115:** PVP\_THCn\_TH12 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

## THCn Threshold Value 13

The PVP\_THCn\_TH13 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH13: THCn Threshold Value 13 - R/W

Reset = 0x7fff ffff

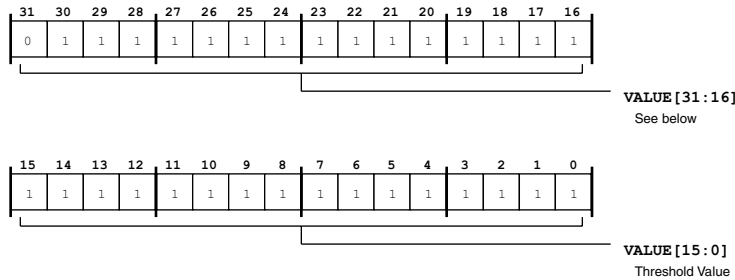


Figure 30-150: PVP\_THCn\_TH13 Register Diagram

Table 30-116: PVP\_THCn\_TH13 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

THCn Threshold Value 14

The PVP\_THCn\_TH14 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH14: THCn Threshold Value 14 - R/W

Reset = 0x7fff ffff

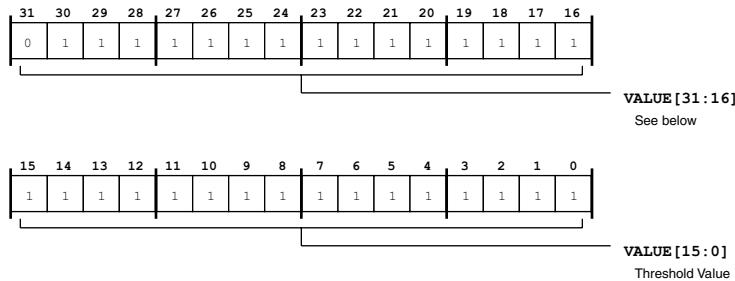


Figure 30-151: PVP\_THCn\_TH14 Register Diagram

Table 30-117: PVP\_THCn\_TH14 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

# THCn Threshold Value 15

The PVP\_THCn\_TH15 holds one of up to 16 threshold values, which the THC uses in quantization mode and hysteresis mode. For more information about these modes and the registers involved in their operation, see the PVP functional description.

PVP\_THCn\_TH15: THCn Threshold Value 15 - R/W

Reset = 0x7fff ffff

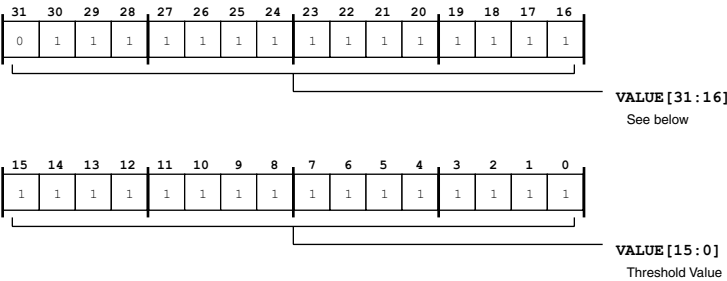


Figure 30-152: PVP\_THCn\_TH15 Register Diagram

Table 30-118: PVP\_THCn\_TH15 Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Threshold Value.        |

# THCn Histogram Horizontal Position

The PVP\_THCn\_HHPOS provides RLE encoder window information that the PVP uses for RLE compression operations.

PVP\_THCn\_HHPOS: THCn Histogram Horizontal Position - R/W

Reset = 0x0000 0000

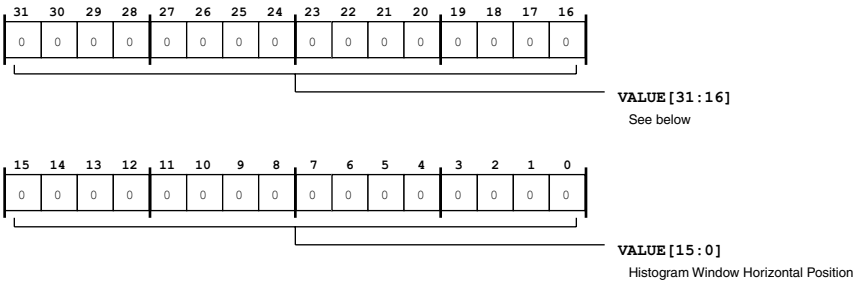


Figure 30-153: PVP\_THCn\_HHPOS Register Diagram

Table 30-119: PVP\_THCn\_HHPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                      |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Histogram Window Horizontal Position.<br>The PVP_THCn_HHPOS.VALUE provides the histogram window start X-coordinate position. |

THCn Histogram Vertical Position

The PVP\_THCn\_HVPOS provides RLE encoder window information that the PVP uses for RLE compression operations.

PVP\_THCn\_HVPOS: THCn Histogram Vertical Position - R/W

Reset = 0x0000 0000

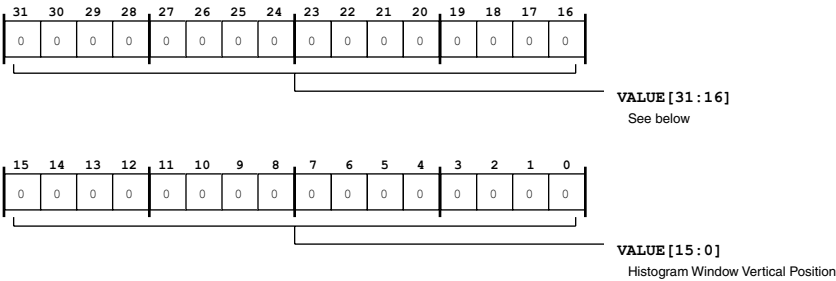


Figure 30-154: PVP\_THCn\_HVPOS Register Diagram

Table 30-120: PVP\_THCn\_HVPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Histogram Window Vertical Position.<br>The PVP_THCn_HVPOS.VALUE provides the histogram window start Y-coordinate position. |

THCn Histogram Horizontal Count

The PVP\_THCn\_HHCNT provides RLE encoder window information that the PVP uses for RLE compression operations.

**PVP\_THCn\_HHCNT: THCn Histogram Horizontal Count - R/W**

Reset = 0x0000 0000

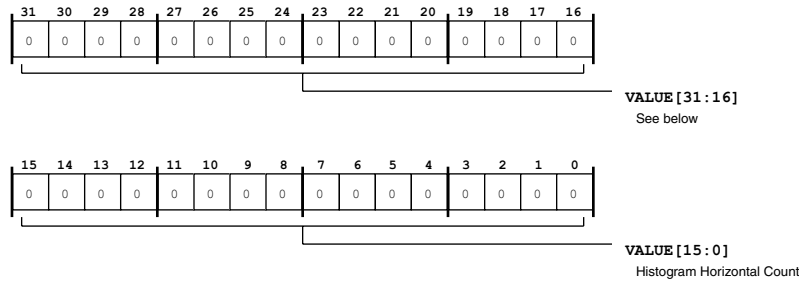


**Figure 30-155:** PVP\_THCn\_HHCNT Register Diagram

**Table 30-121:** PVP\_THCn\_HHCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Histogram Horizontal Count.<br>The PVP_THCn_HHCNT.VALUE provides the histogram window width (count) in the X dimension. |

**THCn Histogram Vertical Count**

The PVP\_THCn\_HVCNT provides RLE encoder window information that the PVP uses for RLE compression operations.

**PVP\_THCn\_HVCNT: THCn Histogram Vertical Count - R/W**

Reset = 0x0000 0000

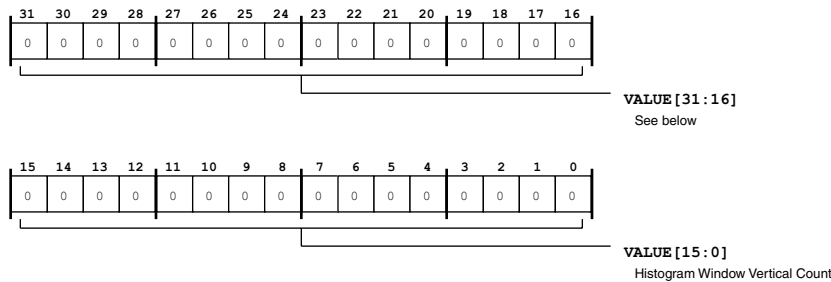


**Figure 30-156:** PVP\_THCn\_HVCNT Register Diagram

Table 30-122: PVP\_THCn\_HVCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                              |
|---------|----------|----------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Histogram Window Vertical Count.<br>The PVP_THCn_HVCNT.VALUE provides the histogram window width in the Y dimension. |

THCn RLE Horizontal Position

The PVP\_THCn\_RHPOS provides RLE encoder window information that the PVP uses for RLE compression operations.

PVP\_THCn\_RHPOS: THCn RLE Horizontal Position - R/W

Reset = 0x0000 0000

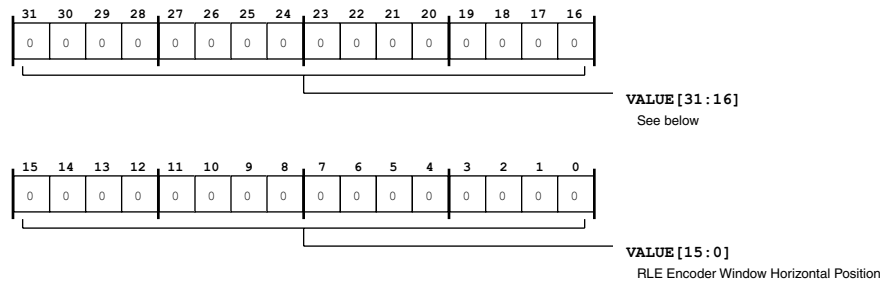


Figure 30-157: PVP\_THCn\_RHPOS Register Diagram

Table 30-123: PVP\_THCn\_RHPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | RLE Encoder Window Horizontal Position.<br>The PVP_THCn_RHPOS.VALUE provides the RLE encoder window start X-coordinate position. |

THCn RLE Vertical Position

The PVP\_THCn\_RVPOS provides RLE encoder window information that the PVP uses for RLE compression operations.

**PVP\_THCn\_RVPOS: THCn RLE Vertical Position - R/W**

Reset = 0x0000 0000

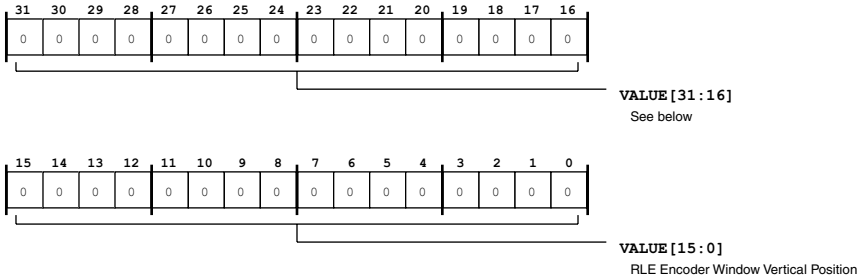


**Figure 30-158:** PVP\_THCn\_RVPOS Register Diagram

**Table 30-124:** PVP\_THCn\_RVPOS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | RLE Encoder Window Vertical Position.<br>The PVP_THCn_RVPOS.VALUE provides the RLE encoder window start Y-coordinate position. |

**THCn RLE Horizontal Count**

The PVP\_THCn\_RHCNT provides RLE encoder window information that the PVP uses for RLE compression operations.

**PVP\_THCn\_RHCNT: THCn RLE Horizontal Count - R/W**

Reset = 0x0000 0000

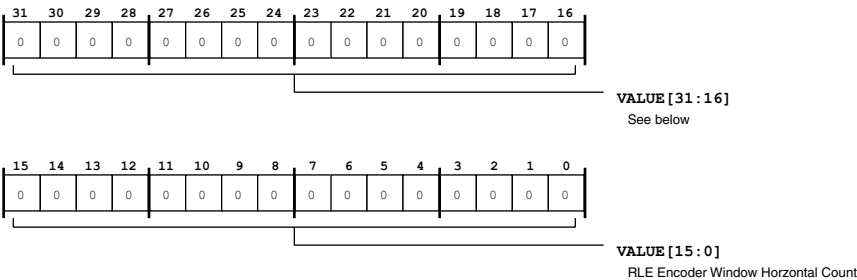


**Figure 30-159:** PVP\_THCn\_RHCNT Register Diagram

Table 30-125: PVP\_THCn\_RHCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | RLE Encoder Window Horizontal Count.<br>The PVP_THCn_RHCNT.VALUE provides the RLE encoder window width (count) in the X dimension. |

THCn RLE Vertical Count

The PVP\_THCn\_RVCNT provides RLE encoder window information that the PVP uses for RLE compression operations.

PVP\_THCn\_RVCNT: THCn RLE Vertical Count - R/W

Reset = 0x0000 0000

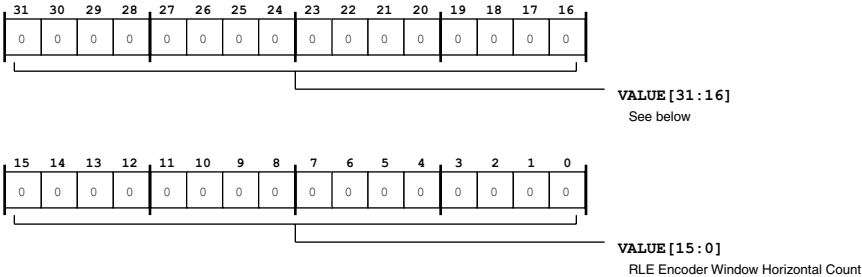


Figure 30-160: PVP\_THCn\_RVCNT Register Diagram

Table 30-126: PVP\_THCn\_RVCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | RLE Encoder Window Horizontal Count.<br>The PVP_THCn_RVCNT.VALUE provides the RLE encoder window width (count) in the Y dimension. |

THCn Histogram Frame Count Status

The PVP\_THCn\_HFCNT\_STAT indicates the current histogram frame count. The number of frames for which the histogram counters operate is determined by the PVP\_THCn\_HFCNT register. On reaching this count, the histogram counters are cleared and restart again.



**PVP\_THCn\_HFCNT\_STAT: THCn Histogram Frame Count Status - R/NW**

Reset = 0x0000 0000

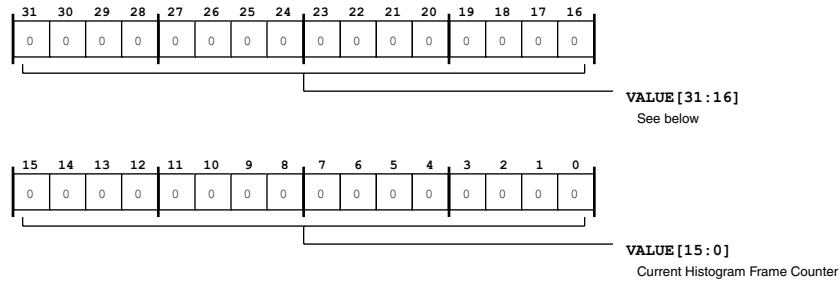


**Figure 30-161:** PVP\_THCn\_HFCNT\_STAT Register Diagram

**Table 30-127:** PVP\_THCn\_HFCNT\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration          |
|---------|----------|----------------------------------|
| 31:0    | VALUE    | Current Histogram Frame Counter. |

## THCn Histogram Counter Value 0

The PVP\_THCn\_HCNT0\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

**PVP\_THCn\_HCNT0\_STAT: THCn Histogram Counter Value 0 - R/NW**

Reset = 0x0000 0000

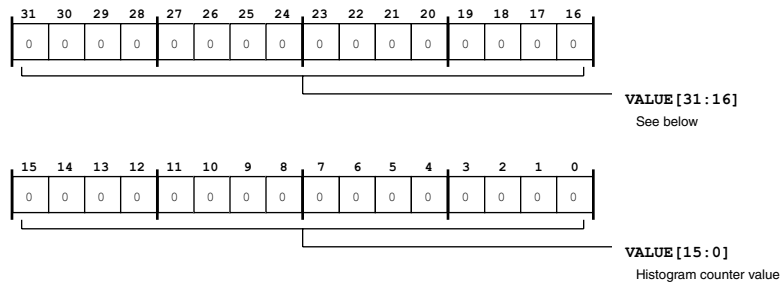


**Figure 30-162:** PVP\_THCn\_HCNT0\_STAT Register Diagram

**Table 30-128:** PVP\_THCn\_HCNT0\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Histogram Counter Value 1

The PVP\_THCn\_HCNT1\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT1\_STAT: THCn Histogram Counter Value 1 - R/NW

Reset = 0x0000 0000

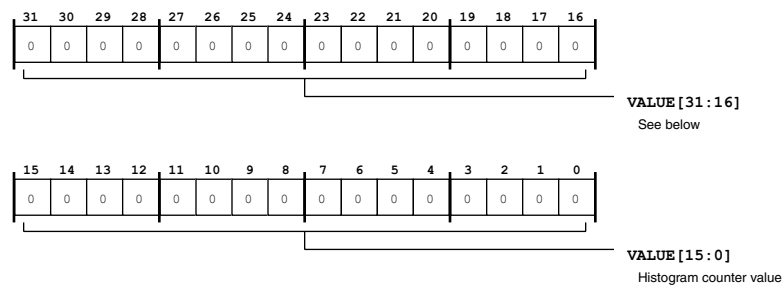


Figure 30-163: PVP\_THCn\_HCNT1\_STAT Register Diagram

Table 30-129: PVP\_THCn\_HCNT1\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Histogram Counter Value 2

The PVP\_THCn\_HCNT2\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT2\_STAT: THCn Histogram Counter Value 2 - R/NW

Reset = 0x0000 0000

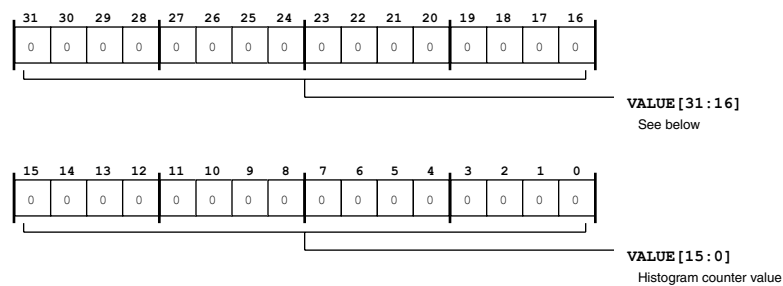


Figure 30-164: PVP\_THCn\_HCNT2\_STAT Register Diagram

**Table 30-130:** PVP\_THCn\_HCNT2\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 3

The PVP\_THCn\_HCNT3\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

**PVP\_THCn\_HCNT3\_STAT: THCn Histogram Counter Value 3 - R/NW**

Reset = 0x0000 0000

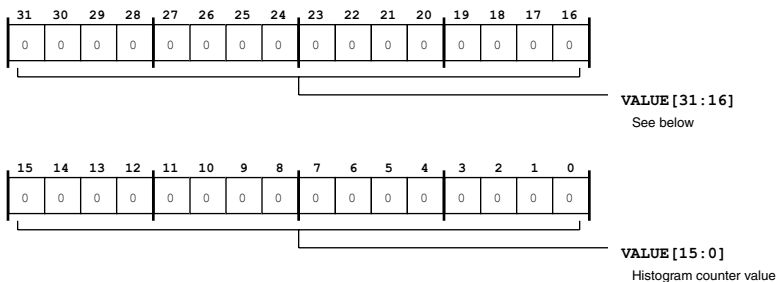


**Figure 30-165:** PVP\_THCn\_HCNT3\_STAT Register Diagram

**Table 30-131:** PVP\_THCn\_HCNT3\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 4

The PVP\_THCn\_HCNT4\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT4\_STAT: THCn Histogram Counter Value 4 - R/NW

Reset = 0x0000 0000

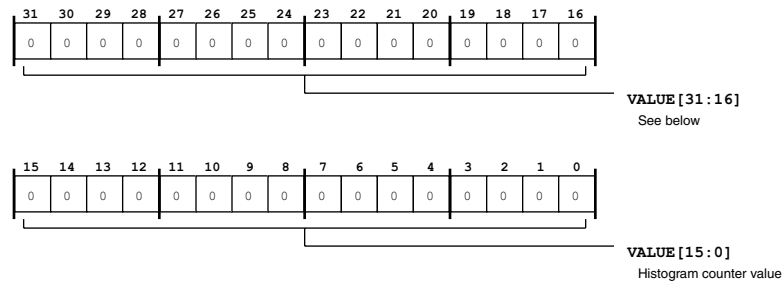


Figure 30-166: PVP\_THCn\_HCNT4\_STAT Register Diagram

Table 30-132: PVP\_THCn\_HCNT4\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

THCn Histogram Counter Value 5

The PVP\_THCn\_HCNT5\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT5\_STAT: THCn Histogram Counter Value 5 - R/NW

Reset = 0x0000 0000

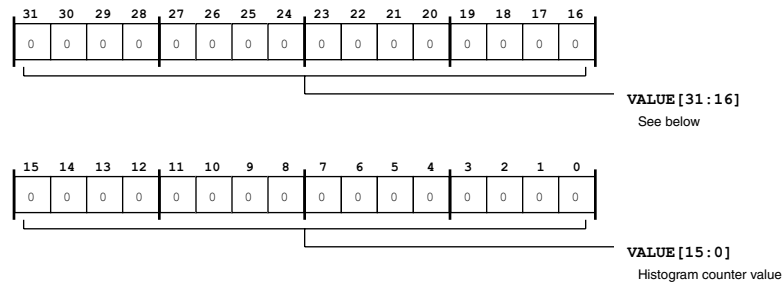


Figure 30-167: PVP\_THCn\_HCNT5\_STAT Register Diagram

Table 30-133: PVP\_THCn\_HCNT5\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 6

The PVP\_THCn\_HCNT6\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT6\_STAT: THCn Histogram Counter Value 6 - R/NW

Reset = 0x0000 0000

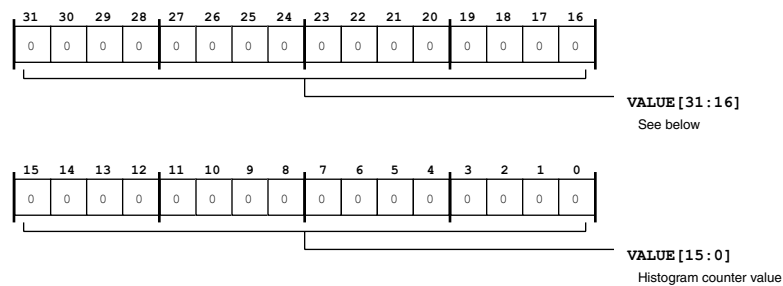


Figure 30-168: PVP\_THCn\_HCNT6\_STAT Register Diagram

Table 30-134: PVP\_THCn\_HCNT6\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 7

The PVP\_THCn\_HCNT7\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT7\_STAT: THCn Histogram Counter Value 7 - R/NW

Reset = 0x0000 0000

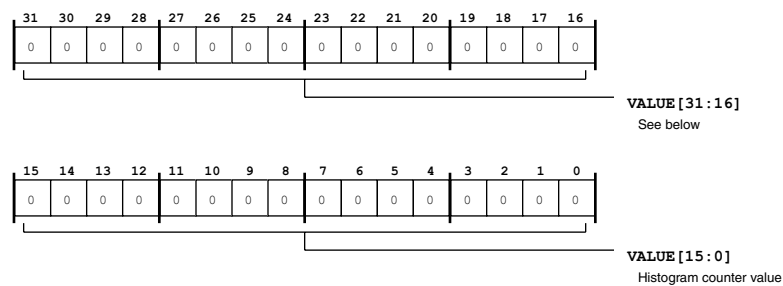


Figure 30-169: PVP\_THCn\_HCNT7\_STAT Register Diagram

Table 30-135: PVP\_THCn\_HCNT7\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

THCn Histogram Counter Value 8

The PVP\_THCn\_HCNT8\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT8\_STAT: THCn Histogram Counter Value 8 - R/NW

Reset = 0x0000 0000

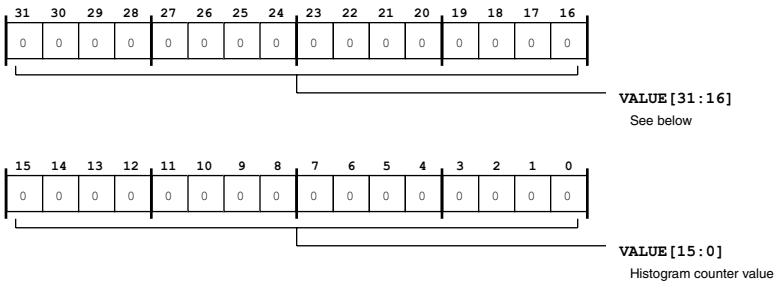


Figure 30-170: PVP\_THCn\_HCNT8\_STAT Register Diagram

Table 30-136: PVP\_THCn\_HCNT8\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

THCn Histogram Counter Value 9

The PVP\_THCn\_HCNT9\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

**PVP\_THCn\_HCNT9\_STAT: THCn Histogram Counter Value 9 - R/NW**

Reset = 0x0000 0000

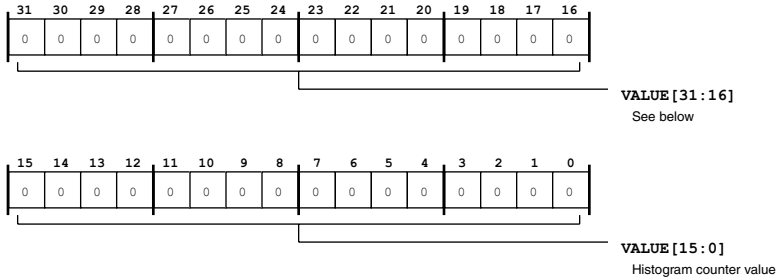


**Figure 30-171:** PVP\_THCn\_HCNT9\_STAT Register Diagram

**Table 30-137:** PVP\_THCn\_HCNT9\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Histogram Counter Value 10

The PVP\_THCn\_HCNT10\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

**PVP\_THCn\_HCNT10\_STAT: THCn Histogram Counter Value 10 - R/NW**

Reset = 0x0000 0000

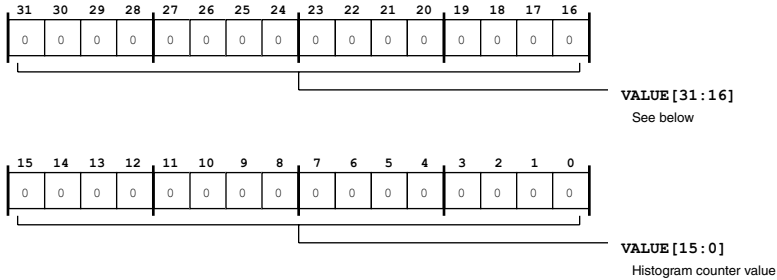


**Figure 30-172:** PVP\_THCn\_HCNT10\_STAT Register Diagram

**Table 30-138:** PVP\_THCn\_HCNT10\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Histogram Counter Value 11

The PVP\_THCn\_HCNT11\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT11\_STAT: THCn Histogram Counter Value 11 - R/NW

Reset = 0x0000 0000

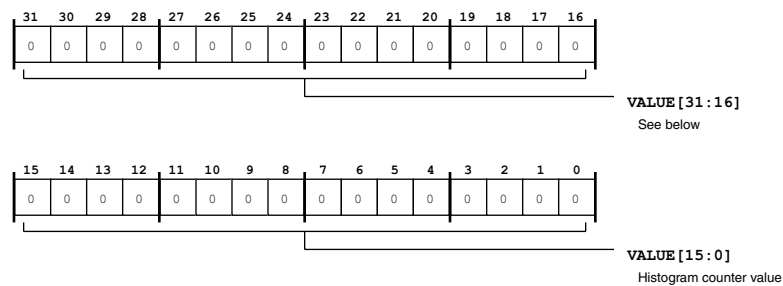


Figure 30-173: PVP\_THCn\_HCNT11\_STAT Register Diagram

Table 30-139: PVP\_THCn\_HCNT11\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Histogram Counter Value 12

The PVP\_THCn\_HCNT12\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT12\_STAT: THCn Histogram Counter Value 12 - R/NW

Reset = 0x0000 0000

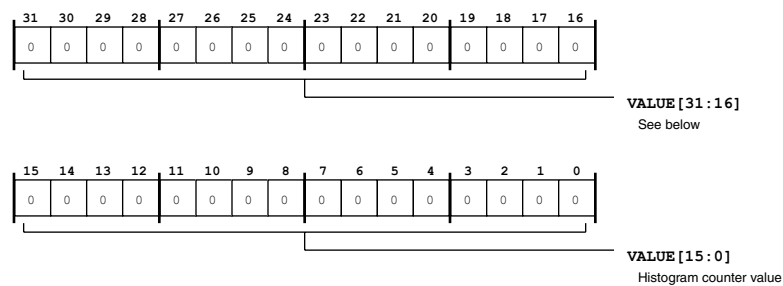


Figure 30-174: PVP\_THCn\_HCNT12\_STAT Register Diagram



**Table 30-140:** PVP\_THCn\_HCNT12\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 13

The PVP\_THCn\_HCNT13\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

**PVP\_THCn\_HCNT13\_STAT: THCn Histogram Counter Value 13 - R/NW**

Reset = 0x0000 0000

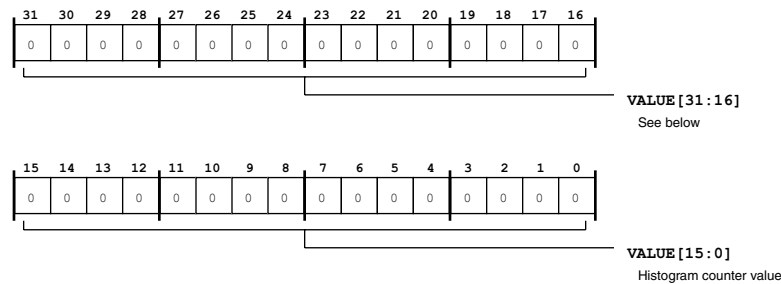


**Figure 30-175:** PVP\_THCn\_HCNT13\_STAT Register Diagram

**Table 30-141:** PVP\_THCn\_HCNT13\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

### THCn Histogram Counter Value 14

The PVP\_THCn\_HCNT14\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT14\_STAT: THCn Histogram Counter Value 14 - R/NW

Reset = 0x0000 0000

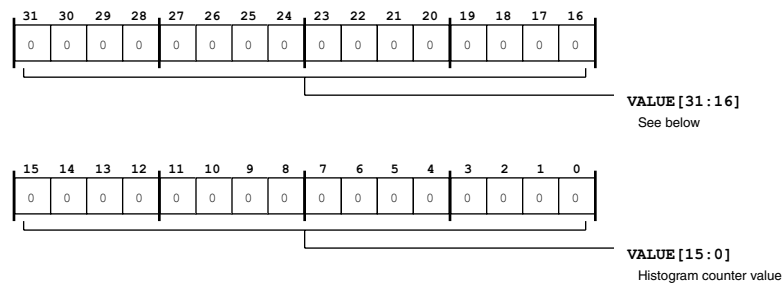


Figure 30-176: PVP\_THCn\_HCNT14\_STAT Register Diagram

Table 30-142: PVP\_THCn\_HCNT14\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

THCn Histogram Counter Value 15

The PVP\_THCn\_HCNT15\_STAT holds the current value for one of the histogram counters. If a histogram counter reaches the maximum value (0xFFFFFFFF) before expiry of histogram frame count (PVP\_THCn\_HFCNT), the histogram counters do not overflow, but saturate at that value.

PVP\_THCn\_HCNT15\_STAT: THCn Histogram Counter Value 15 - R/NW

Reset = 0x0000 0000

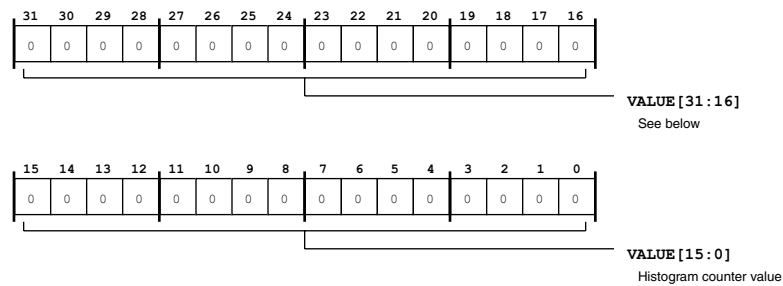


Figure 30-177: PVP\_THCn\_HCNT15\_STAT Register Diagram

Table 30-143: PVP\_THCn\_HCNT15\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration  |
|---------|----------|--------------------------|
| 31:0    | VALUE    | Histogram counter value. |

## THCn Number of RLE Reports

The PVP\_THCn\_RREP\_STAT holds the number of RLE reports. The PVP generates the run length reports at the output of the module as soon as they are available. Note that there is a variable number of reports generated per line or per frame, depending on the pixel patterns.

PVP\_THCn\_RREP\_STAT: THCn Number of RLE Reports - R/NW

Reset = 0x0000 0000

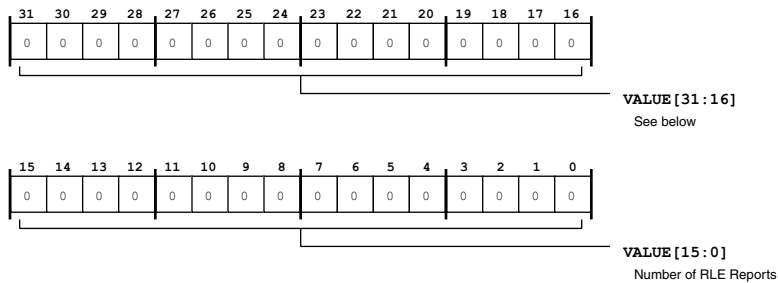


Figure 30-178: PVP\_THCn\_RREP\_STAT Register Diagram

Table 30-144: PVP\_THCn\_RREP\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | VALUE    | Number of RLE Reports.  |

## PMA Configuration

The PVP\_PMA\_CFG register controls the pixel magnitude and angle computation unit (PMA) block's pipeline features, including port selection and block enable.

**PVP\_PMA\_CFG: PMA Configuration - R/W**

Reset = 0x0000 0000

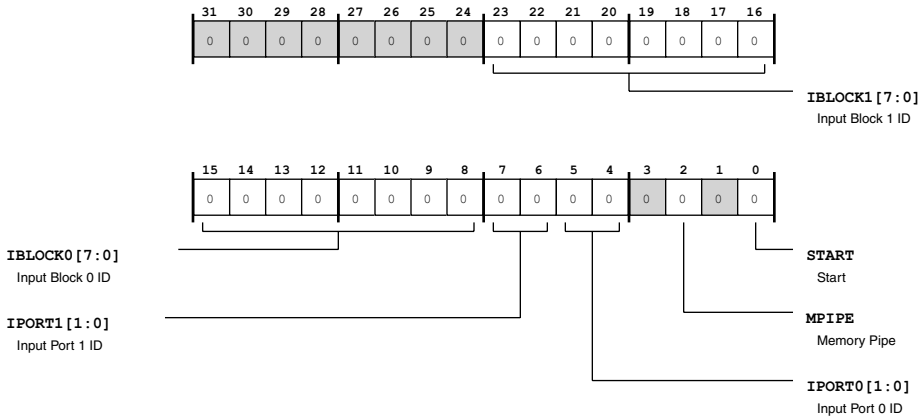


**Figure 30-179:** PVP\_PMA\_CFG Register Diagram

**Table 30-145:** PVP\_PMA\_CFG Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 23:16   | IBLOCK1  | Input Block 1 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PMA_CFG.IBLOCK1 and PVP_PMA_CFG.IPORT1 fields determine which output port(s) are connected to the input port(s). |
|         |          | 12 IPF0                                                                                                                                                                                                                   |
|         |          | 14 IPF1                                                                                                                                                                                                                   |
|         |          | 16 CNV0                                                                                                                                                                                                                   |
|         |          | 20 CNV1                                                                                                                                                                                                                   |
|         |          | 24 CNV2                                                                                                                                                                                                                   |
|         |          | 28 CNV3                                                                                                                                                                                                                   |

**Table 30-145:** PVP\_PMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | IBLOCK0  | Input Block 0 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PMA_CFG.IBLOCK0 and PVP_PMA_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s).                                                                                                              |
|         |          | 12 IPF0                                                                                                                                                                                                                                                                                                                                |
|         |          | 14 IPF1                                                                                                                                                                                                                                                                                                                                |
|         |          | 16 CNV0                                                                                                                                                                                                                                                                                                                                |
|         |          | 20 CNV1                                                                                                                                                                                                                                                                                                                                |
|         |          | 24 CNV2                                                                                                                                                                                                                                                                                                                                |
|         |          | 28 CNV3                                                                                                                                                                                                                                                                                                                                |
| 7:6     | IPORT1   | Input Port 1 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PMA_CFG.IBLOCK1 and PVP_PMA_CFG.IPORT1 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 5:4     | IPORT0   | Input Port 0 ID.<br>The blocks comprising the PVP pipe may connect to one or more output ports. The PVP_PMA_CFG.IBLOCK0 and PVP_PMA_CFG.IPORT0 fields determine which output port(s) are connected to the input port(s). For more information, see the connectivity tables for each input block in the architectural concepts section. |
| 2       | MPIPE    | Memory Pipe.<br>The PVP_PMA_CFG.MPIPE bit selects memory pipe or camera pipe.                                                                                                                                                                                                                                                          |
|         |          | 0 Camera Pipe                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Memory Pipe                                                                                                                                                                                                                                                                                                                          |

**Table 30-145:** PVP\_PMA\_CFG Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                      |                            |
|---------|----------|----------------------------------------------------------------------------------------------|----------------------------|
| 0       | START    | Start.<br>The PVP_PMA_CFG.START enables the PVP. This bit reads 0 after the pipe is drained. |                            |
|         |          | 1                                                                                            | Write to Enable the Module |

# 31 Enhanced Parallel Peripheral Interface (EPPI)

The enhanced parallel peripheral interface (EPPI) is a half-duplex, bidirectional port with a dedicated clock pin and three frame sync (FS) pins. It can support direct connections to active TFT LCD, parallel A/D and D/A converters, video encoders and decoders, image sensor modules and other general-purpose peripherals. Each EPPI has a DMA channel associated with it. Moreover, in some modes, a EPPI may use an additional DMA channel.

## EPPI Features

The following features are supported in the EPPI module.

- Programmable data length: 8, 10, 12, 14, 16, 18 and 24 bits per clock cycle
- Bidirectional and half-duplex port
- Internal or external clock source
- Clock gating by an external device asserting the clock gating control signal
- Various framed and non-framed operating modes as well as internal or external frame syncs
- Various general-purpose modes with 1, 2, 3 and 0 frame sync modes for both receive and transmit
- Ignores premature external frame syncs for data consistency
- SMPTE274M and SMPTE 296M high definition format support
- ITU-656, SMPTE 296M and SMPTE 274M status word error detection and correction for ITU-656 receive modes (at sampling frequencies less than or equal to 74.25 MHz)
- ITU-656 receive modes – active video only, vertical blanking only, and entire field
- ITU-656, SMPTE 296M and SMPTE 274M preamble and status word decode (at sampling frequencies less than or equal to 74.25 MHz)
- Optional packing and unpacking of data to/from 32 bits from/to 8, 16 and 24 bits. If packing/unpacking is enabled, endianness can be altered to change the order of packing/unpacking of bytes/words
- Optional sign extension or zero-fill and alternate even or odd data sample filter for receive modes
- RGB888 to RGB666 or RGB565 conversion for transmit modes
- 4:2:2 YCrCb data Tx/Rx interleaving/de-interleaving modes

- Configurable LCD data enable (DEN) output available on frame sync 3
- Supports delayed start of PPI frame syncs
- Data clipping and mirroring
- Horizontal and vertical windowing for general-purpose 2 and 3 frame sync modes
- Preamble, blanking and stripping support
- Multiplexing dual input

## EPPI Functional Description

The following sections provide a functional descriptions of the EPPI.

### RGB Data Formats

For transmit modes, the EPPI can convert RGB888 data in memory to RGB666 at the output if the `RGBFMTEN` bit is set in the `EPPI_CTL` register and if `DLEN` is equal to 18 bits. Similarly, the EPPI can convert RGB888 data in memory to RGB565 at the output if the `RGBFMTEN` bit is set in the `EPPI_CTL` register and if `DLEN` is equal to 16 bits.

This conversion is done as follows: if `PACKEN` = 1, the EPPI first unpacks, according to the `SWAPEN` settings, the three 32-bit data words from the DMA into four 24-bit data words to be transmitted out as described earlier. If `PACKEN` = 0, then the EPPI takes the lower 24 bits of the 32-bit DMA as the data to be transmitted. Then the EPPI truncates this 24-bit data word to the required data width by removing the lower 2 bits of G and the lower 2 or 3 bits of R and B.

### Data Clipping

The EPPI contains two registers to define the lower and upper limits for the Luma and Chroma components. This is used for clipping data values during 8, 10, 12 or 16-bit transmit modes. All data values for odd samples which are less than the value in the `LOW_ODD` field in the `EPPI_ODDCLIP` register are replaced with the value in the `LOW_ODD` field and all data values for even samples which are less than the value in the `LOW_EVEN` field in the `EPPI_EVENCLIP` register are replaced with the value in the `LOW_EVEN` field.

In the same manner, all data values for odd samples which are more than the value in the `HIGH_ODD` field in the `EPPI_ODDCLIP` register are replaced with the value in the `HIGH_ODD` field and all data values for even samples which are more than the values in the `HIGH_EVEN` field are replaced with the values in the `HIGH_EVEN` field. Depending upon the programmed EPPI length, only the corresponding bits (least aligned) are considered for clipping. For example if the EPPI is programmed in 10-bit mode, bits 9–0 and bits 25–16 constitute the clipping thresholds. The higher bits are ignored. The EPPI supports 8, 10, 12, and 16-bit clipping thresholds.



For the 4:2:2 YCrCb color space, Luma and Chroma typically have different lower and upper thresholds, which is why separate thresholds may be required for even and odd data samples. In the case of mono-chrome (Y only) or some non-video clipping applications, the value in the LOW\_ODD field should be the same as the value in the LOW\_EVEN field, and the value in the HIGH\_ODD field should be the same as the value in the HIGH\_EVEN field.

In GP 0 FS mode with internal blanking generation, clipping is valid only for the active video part of the transmitted data. ITU-R 656 preambles, status words and blanking data bypass the clipping logic.

If the EPPI is programmed in 16, 20 or 24-bit mode with SPLTWRD bit set, the YDATA (luma data) gets the clipping threshold levels of EPPI\_EVENCLIP and the CDATE (chroma data) gets the clipping threshold levels of EPPI\_ODDCLIP.

The clipping registers are ignored when the RGBFMTEN bit is set in the EPPI\_CTL register.

## Data Mirroring

To increase the pin multiplexing options with respect to the EPPI data pins, an additional data mirroring feature is available which mirrors the EPPI data bits 15–0. The MMR bit (DMRR) if set, mirrors the bits (default is DMRR = 0, no mirror). The complete description of this bit is provided in EPPI\_CTL register section. This feature is available in both transmit and receive.

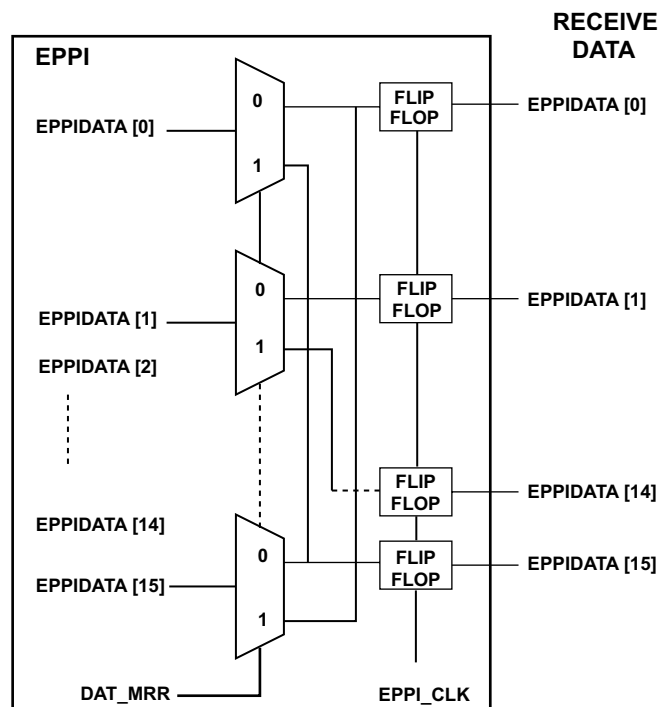


**Figure 31-1:** Data Mirroring Receive

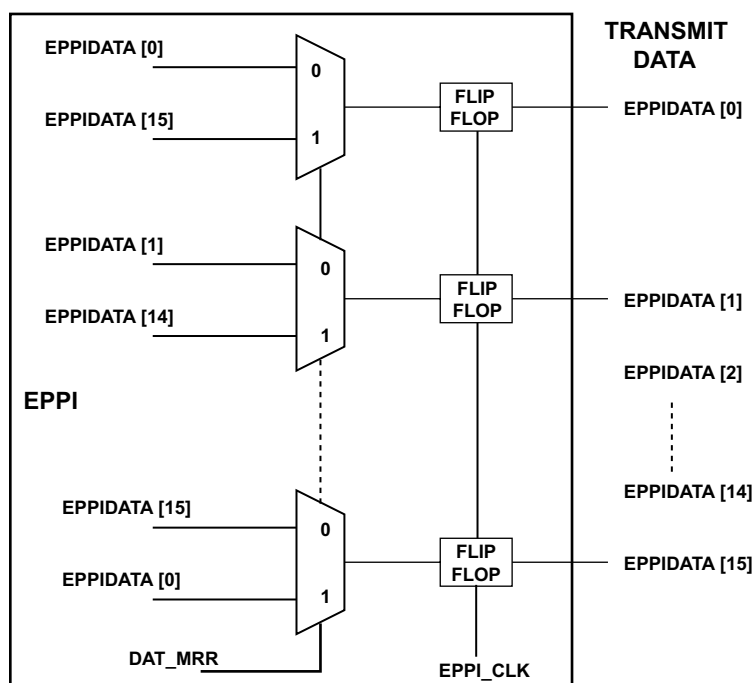


**Figure 31-2:** Data Mirroring Transmit

## Windowing

The EPPI supports windowing for GP input modes. The module can be configured to bring in a region of interest instead of the entire frame of data. This helps reduce bandwidth.

## Preamble, Blanking and Stripping Support

The module supports embedding blanking information and clipping active data to be transmitted. However this is limited to either single channel data or interleaved data (for example Y alone, C alone, or Y-Cr/Cb interleaved). Adding blanking information or clipping the data is not available for data sent in parallel [Y:Cr/Cb] for  $DLEN = 20$  or  $24$  bits, as would be the case with SMPTE modes. To support all the SMPTE modes, this logic is extended to  $20$  and  $24$  bits. The additional logic goes into the receive (preamble correction, sign extension) and transmit (blankgen and clipping) modes. This feature is available for  $DLEN = 16$  bits as well.

The EPPI supports preamble generation/detection and blank stripping for the two HD formats SMPTE 274M and 294M. The SMPTE standards are shown below in the table as a comparison with the ITU 656 modes which are currently supported. Preambles for SMPTE and ITU modes are identical. Extension of preamble to  $12$  bits is also supported.

**Table 31-1:** Video Mode Comparison

| Video Mode     | Frame Rate | Frame Resolution | Active Video Resolution | Sampling Frequency (MHz) | Remarks         |
|----------------|------------|------------------|-------------------------|--------------------------|-----------------|
| ITU 656 (NTSC) | 30         | 1716x525         | 720x480                 | 27.2                     | Y-C interleaved |
| ITU 656 (PAL)  | 25         | 1728x625         | 720x576                 | 26.8                     | Y-C interleaved |
| SMPTE 296M     | 30         | 3300x750         | 1280x720                | 74.25                    | Y,C separate    |
|                | 60         | 1650x750         | 1280x720                | 74.25                    | Y,C separate    |
| SMPTE 274M     | 30         | 2200x1125        | 1920x1080               | 74.25                    | Y,C separate    |
|                | 60         | 2200x1125        | 1920x1080               | 148.5                    | Y,C separate    |
|                | 25         | 2640x1125        | 1920x1080               | 74.25                    | Y,C separate    |
|                | 50         | 2640x1125        | 1920x1080               | 148.5                    | Y,C separate    |
|                | 24         | 2750x1125        | 1920x1080               | 74.25                    | Y,C separate    |

The EPPI supports all SMPTE modes shown above which run at a sampling frequency not higher than 74.25 MHz.

## EPPI Register List

The EPPI is a half-duplex, bidirectional port accommodating up to 16 bits of data. It has a dedicated clock pin and three multiplexed frame sync pins. The highest system throughput is achieved with 8-bit data, since two 8-bit data samples can be packed as a single 16-bit word. In such a case, the earlier sample is placed in the 8 least significant bits (LSBs). For more information on EPPI functionality, see the EPPI register descriptions.

**Table 31-2:** EPPI Module Registers

| Name      | Description                        |
|-----------|------------------------------------|
| EPPI_STAT | Status Register                    |
| EPPI_HCNT | Horizontal Transfer Count Register |
| EPPI_HDLY | Horizontal Delay Count Register    |

**Table 31-2:** EPPI Module Registers (Continued)

| Name           | Description                                                             |
|----------------|-------------------------------------------------------------------------|
| EPPI_VCNT      | Vertical Transfer Count Register                                        |
| EPPI_VDLY      | Vertical Delay Count Register                                           |
| EPPI_FRAME     | Lines Per Frame Register                                                |
| EPPI_LINE      | Samples Per Line Register                                               |
| EPPI_CLKDIV    | Clock Divide Register                                                   |
| EPPI_CTL       | Control Register                                                        |
| EPPI_FS1_WLHB  | FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register |
| EPPI_FS1_PASPL | FS1 Period Register / EPPI Active Samples Per Line Register             |
| EPPI_FS2_WLVB  | FS2 Width Register / EPPI Lines Of Vertical Blanking Register           |
| EPPI_FS2_PALPF | FS2 Period Register / EPPI Active Lines Per Field Register              |
| EPPI_IMSK      | Interrupt Mask Register                                                 |
| EPPI_ODDCLIP   | Clipping Register for ODD (Chroma) Data                                 |
| EPPI_EVENCLIP  | Clipping Register for EVEN (Luma) Data                                  |
| EPPI_FS1_DLY   | Frame Sync 1 Delay Value                                                |
| EPPI_FS2_DLY   | Frame Sync 2 Delay Value                                                |
| EPPI_CTL2      | Control Register 2                                                      |

## ADSP-BF60x EPPI Interrupt List

**Table 31-3:** ADSP-BF60x EPPI Interrupt List

| Description         | ID | DMA Channel | Sensitivity |
|---------------------|----|-------------|-------------|
| EPPI0 Channel 0 DMA | 98 | 29          | LEVEL       |
| EPPI0 Channel 1 DMA | 99 | 30          | LEVEL       |

**Table 31-3:** ADSP-BF60x EPPI Interrupt List (Continued)

| Description         | ID  | DMA Channel | Sensitivity |
|---------------------|-----|-------------|-------------|
| EPPI0 Status        | 100 |             | LEVEL       |
| EPPI2 Channel 0 DMA | 101 | 31          | LEVEL       |
| EPPI2 Channel 1 DMA | 102 | 32          | LEVEL       |
| EPPI2 Status        | 103 |             | LEVEL       |
| EPPI1 Channel 0 DMA | 104 | 33          | LEVEL       |
| EPPI1 Channel 1 DMA | 105 | 34          | LEVEL       |
| EPPI1 Status        | 106 |             | LEVEL       |

## ADSP-BF60x EPPI Trigger List

**Table 31-4:** Masters ADSP-BF60x EPPI Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| EPPI0 Channel 0 DMA | 51 | PULSE/EDGE  |
| EPPI0 Channel 1 DMA | 52 | PULSE/EDGE  |
| EPPI2 Channel 0 DMA | 53 | PULSE/EDGE  |
| EPPI2 Channel 1 DMA | 54 | PULSE/EDGE  |
| EPPI1 Channel 0 DMA | 55 | PULSE/EDGE  |
| EPPI1 Channel 1 DMA | 56 | PULSE/EDGE  |
|                     |    |             |

**Table 31-5:** Slaves ADSP-BF60x EPPI Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| EPPI0 Channel 0 DMA | 51 |             |

**Table 31-5:** Slaves ADSP-BF60x EPPI Trigger List (Continued)

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| EPPI0 Channel 1 DMA | 52 |             |
| EPPI2 Channel 0 DMA | 53 |             |
| EPPI2 Channel 1 DMA | 54 |             |
| EPPI1 Channel 0 DMA | 55 |             |
| EPPI1 Channel 1 DMA | 56 |             |
|                     |    |             |

## EPPI Definitions

### ITU-R BT. 656

Description of a digital video protocol for interfaces and data stream format required to send uncompressed PAL or NTSC standard definition TV (525 or 625 lines) signals.

### YUV422

YUV is a color space where pixels are defined by a luminance (Y) component and chrominance (UV) components. The suffix signifies how the chrominance components have been decimated as well as formatting. In this case, the YUV422 format has the chrominance decimated by two, meaning only half of each chrominance component are available. Typical YUV422 formatting interleaves the luminance and chrominance as such as U1Y1V1Y2U2Y3V2Y4.

### RGB888

RGB is a color space where pixels are defined by three color values; one red (R), one green (G) and one blue (B). The suffix signifies the bit widths for these color components. In this case, RGB888 means that each red, green and blue value is 8-bits each.

### RGB565

RGB is a color space where pixels are defined by three color values; one red (R), one green (G) and one blue (B). The suffix signifies the bit widths for these color components. In this case, RGB565 means that

the red (R) and blue (B) are 5-bits each while the green (G) is 6-bits. When packed together, each RGB565 pixel can be represented in a 16-bit data word. This format is commonly used in LCD display panels.

### SMPTE 274M

An HD standard defining the spatial resolution (image sample structure) and frame rates for 1920x1080.

### SMPTE 296M

An HD standard for defining the spatial resolution (image sample structure) and frame rates from 1280x720.

## EPPI Block Diagram

The figure shows the functional blocks within the EPPI.

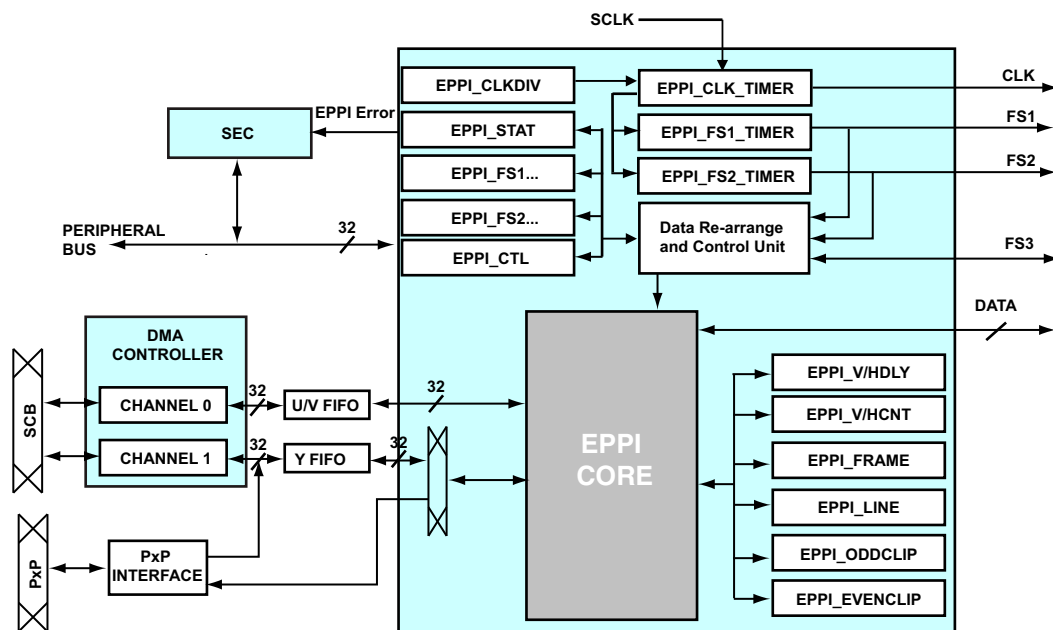


Figure 31-3: EPPI Block Diagram

## EPPI Architectural Concepts

This section describes EPPI architectural concepts, such as EPPI interface, reset operations frame syncs and sampling, DMA, PxP interface, and EPPI clock.

## EPPI Interface

The EPPI has a pixel pipe (PxP) interface in addition to the interface to the system crossbar. In transmit, the PxP interface is in the SCLK0 domain while in receive mode, the PxP interface is in the EPPI clock domain.

A block diagram of the architecture for the EPPI interface is shown below

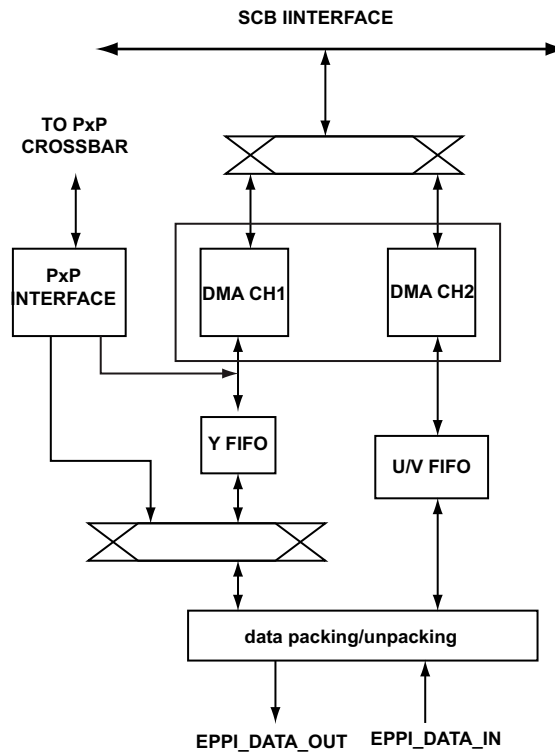


**Figure 31-4:** EPPI DMA and PxP Interface

If the EPPI is configured for transmit, the Y channel data path can take data from either the YFIFO or the PxP interface. When configured to take data from the PxP interface, this PxP data is transmitted on the EPPI pins in all modes. When programmed to take data from the PxP, the DMACFG must be programmed as 0. Luma (and chroma) component should be made available only through the PxP interface. The control signals for deciding from which channel (DMA or PxP) the data is to be taken, is made available to the EPPI by the video subsystem block. When the EPPI is configured for PxP transmit, the incoming data and valid from the PxP are sampled on the SCLK and the outgoing Ready signal is driven on SCLK.

If the EPPI is configured for receive, the EPPI data can be transferred to either the PxP interface or the YFIFO or both. Here only the data going to DMA channel 1 is sent to the PxP. This could be either the Y data only (in split receive mode with EPPI\_CTL.DMACFG set) or the full EPPI input pin data in any other mode. If the EPPI is connected to the PIXC via the PxP, the EPPI can be programmed in a single DMA mode (EPPI\_CTL.DMACFG not set) so that the entire data from the pins of the EPPI flow through the YFIFO (dma channel 1). The control signals for deciding to which channel (DMA or PxP or to both) the data is to be transferred, is made available to the EPPI by the video subsystem block.



## Reset Operation

On a hardware reset, the entire EPPI is reset. All MMRs return to their default values. EPPI interrupt and DMA requests go inactive. Internally generated EPPI\_CLK and frame syncs are aborted.

In software, the EPPI can be reset and re-configured by writing 0 to the EN bit in the EPPI\_CTL register. When disabled in this manner, only the EPPI\_STAT register is cleared to its reset value and interrupts and DMA requests go inactive, and internally generated clock and frame syncs are aborted.

## Frame Sync Polarity and Sampling Edge

The POLS and POLC bits provide a mechanism to select the active level of the frame syncs and the sampling/driving edge of the EPPI clock, respectively. This allows the EPPI to connect to data sources and receivers with a wide array of control signal polarities. Often, the remote data source/receiver also offers configurable signal polarities; in these cases, the POLS and POLC bits add flexibility.

**Table 31-6:** Different Settings for POLS

|             | Frame Sync 2                | Frame Sync 1                |
|-------------|-----------------------------|-----------------------------|
| POLS = b#00 | Active high/ starts out low | Active high/ starts out low |
| POLS = b#01 | Active high/ starts out low | Active low/ starts out high |
| POLS = b#10 | Active low/ starts out high | Active high/ starts out low |
| POLS = b#11 | Active low/ starts out high | Active low/ starts out high |

**Table 31-7:** Different Settings for POLC

|                                            | RX           | TX                 |
|--------------------------------------------|--------------|--------------------|
| Drive Data Sample/Drive syncs              | Sample Data  | Sample/drive syncs |
| POLC = b#00<br>Rising edge<br>Rising edge  | Falling edge | Falling edge       |
| POLC = b#01<br>Rising edge<br>Falling edge | Falling edge | Rising edge        |
| POLC = b#10<br>Falling edge<br>Rising edge | Rising edge  | Falling edge       |

**Table 31-7:** Different Settings for POLC (Continued)

|              | RX          | TX          |
|--------------|-------------|-------------|
| POLC = b#11  | Rising edge | Rising edge |
| Falling edge |             |             |
| Falling edge |             |             |

## DMA

The EPPI has a native DMA controller with two channels. A local arbiter arbitrates between these channels. This arbitrated request is forwarded to the system crossbar. The EPPI would have only a single connection to the fabric.

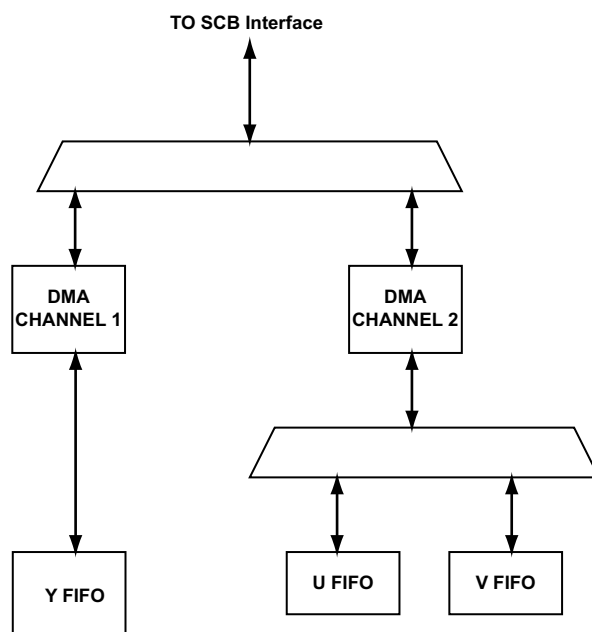


**Figure 31-5:** EPPI DMA Interface

The EPPI must be used with DMA and configuring the EPPI DMA channels is a necessary step toward using the EPPI interface. The channels can be configured for either transmit or receive operation, and have a maximum throughput of  $(EPPI\_CLK) \times (32 \text{ bits/transfer})$ . In modes where data lengths permit, packing may be possible in order to increase transfer bandwidth. The highest throughput is achieved with 8-bit data and packing enabled.

The DMA engine generates interrupts at the completion of a row, frame, or partial-frame transfer. The DMA engine also coordinates the source or destination point for the data that is transferred through the EPPI.

The 2D DMA capability allows the processor to be interrupted at the end of a line or after a frame of video is transferred, or if a DMA error occurs. The `DMA_XCNT` and `DMA_YCNT` MMRs allows for flexible data inter-

rupt points. For example, assume the DMA registers  $\text{DMA\_XMOD} = \text{DMA\_YMOD} = 1$ . If a data frame contains  $320 \times 240$  bytes (240 rows of 320 bytes each), the following conditions hold.

- Setting  $\text{DMA\_XCNT} = 320$ ,  $\text{DMA\_YCNT} = 240$ , and  $\text{DISEL} = 1$  (the  $\text{DISEL}$  bit is located in  $\text{DMA\_CFG}$  register) interrupts on every row transferred, for the entire frame.
- Setting  $\text{DMA\_XCNT} = 320$ ,  $\text{DMA\_YCNT} = 240$ , and  $\text{DISEL} = 0$  interrupts only on the completion of the frame (when 240 rows of 320 bytes have been transferred).
- Setting  $\text{DMA\_XCNT} = 38,400$  ( $320 \times 120$ ),  $\text{DMA\_YCNT} = 2$ , and  $\text{DISEL} = 1$  causes an interrupt when half of the frame is transferred, and again when the whole frame is transferred.

The following is the general procedure for setting up DMA operation with the EPPI.

1. Configure the DMA registers as appropriate for the desired DMA operating mode.
2. Enable the DMA channel for operation.
3. Configure appropriate EPPI registers.
4. Enable the EPPI by writing a 1 to bit 0 in  $\text{EPPI\_CTL}$ .

## Pixel Pipe Interface (PxP)

The EPPI has a pixel pipe (PxP) interface in addition to the SCB interface to the fabric. In transmit, the PxP interface is in the SCLK domain while in receive, the interface is in the EPPI clock domain.

If the EPPI is configured for transmit, the Y channel data path can take data from either the YFIFO or the PxP Crossbar. When configured to take the data from PxP crossbar, this PxP data is transmitted on the EPPI pins in all modes. When programmed to take data from the PxP, the  $\text{DMACFG}$  register must be programmed to 0. Luma (and Chroma) components should be made available only through PxP. The video subsystem blocks (VSS) determine the control signals for deciding the channel (DMA or PxP) from which the data is taken. Refer to the video subsystem section for details.

When the EPPI is configured for PxP transmit, the incoming data and valid signal from the PxP are sampled on the SCLK and the outgoing ready signal is driven on SCLK. This implies the PIXC has to drive out the data in SCLK. The EPPI block with the pixel pipe addition is shown below.

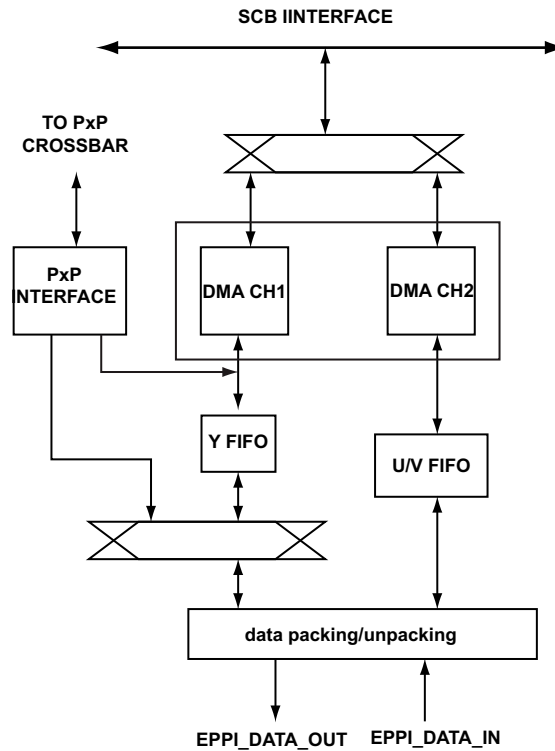


**Figure 31-6:** EPPI Block with the Pixel Pipe Addition

If the EPPI is configured for receive, the EPPI data can be transferred to either the PxP crossbar or the YFIFO or both. Here, only the data going to DMA channel 1 is sent to the PxP. This could be either the Y data only (in split receive mode with  $DMACFG = 1$ ) or the full EPPI input pin data in any other mode. If the EPPI is connected to the PIXC via the PxP, the EPPI can be programmed in a single DMA mode ( $DMACFG = 0$ ) so that the entire data from the pins of the EPPI flows through the Y FIFO (DMA channel 1). The control signals for deciding to which channel (DMA or PxP or to both) the data is to be transferred, is made available to the EPPI by video subsystems block. Refer to the video subsystem section for details.

The following figures describe the design level details of how the PxP interface is multiplexed with the existing data path in transmit and receive case.

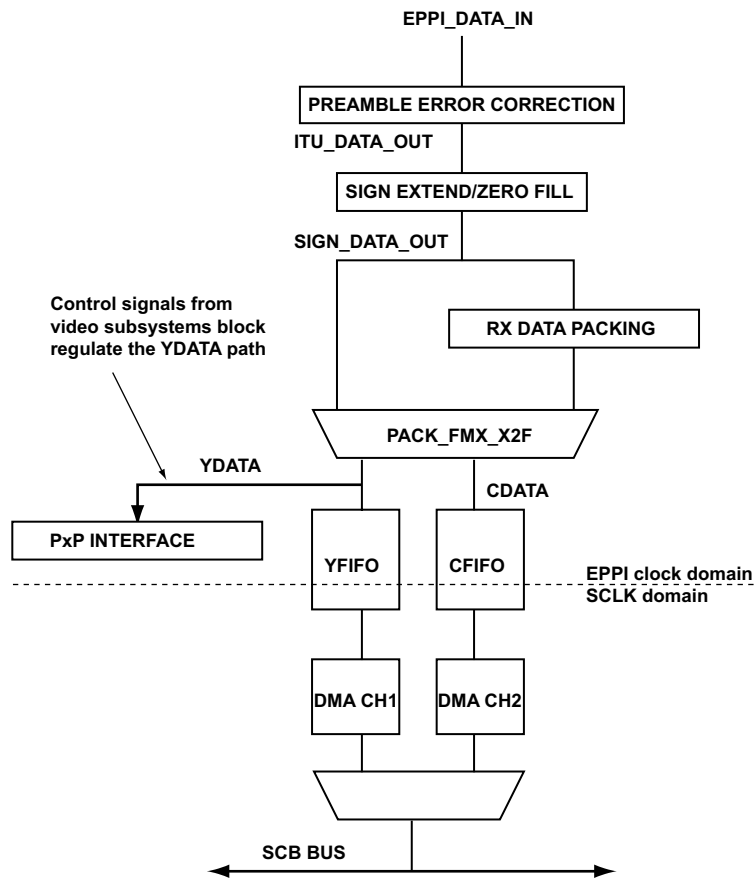


**Figure 31-7:** Receive Data Path with PxP Interface Muxing

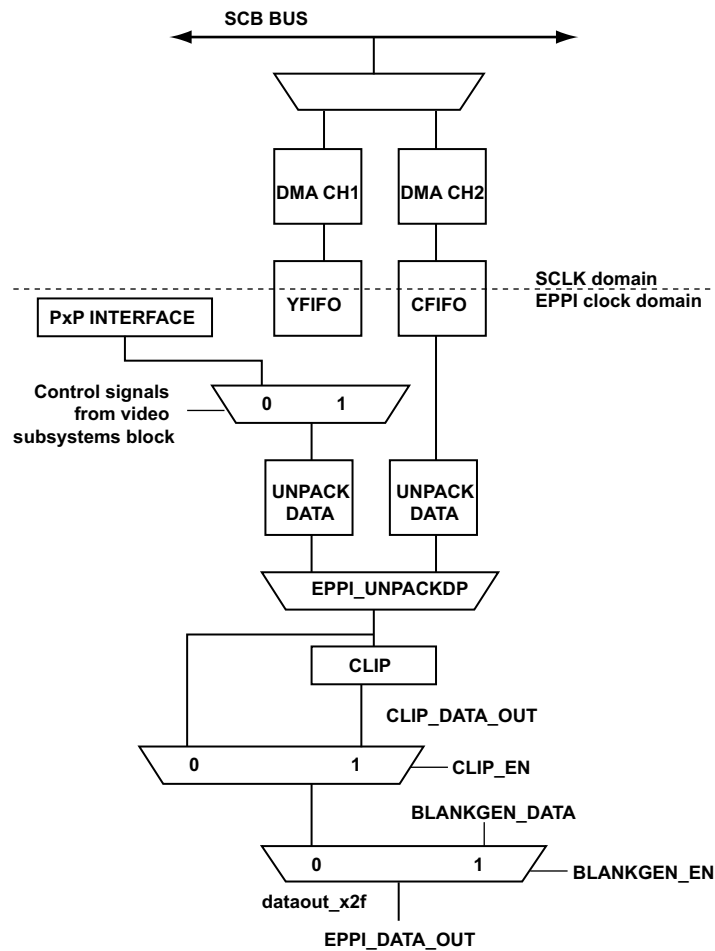


**Figure 31-8:** Transmit Data Path with PxP Interface Muxing

## EPPI Clock

The EPPI can be supplied with an external clock, or the clock can be generated internally and supplied to external devices. When using the internal clock, the maximum frequency possible for EPPI\_CLK is SCLK/2. When using an external clock, the maximum frequency for EPPI\_CLK is 75 MHz.

When using an external EPPI\_CLK, there may be up to two cycles latency before valid data is received or transmitted.

The internal clock can be generated from SCLK if EPPI\_CTL.ICLKGEN is set. The generated clock frequency is then determined by the value in the EPPI\_CLKDIV register. The internally generated EPPI clock frequency is:

$$f_{PCLK} = f_{SCLK} / (CLKDIV + 1)$$

where:

$f_{PCLK}$  – frequency of internally generated EPPI Clock

$f_{\text{SCLK}}$  – frequency of SCLK

CLKDIV - Clock division value programmed in the EPPI\_CLKDIV register.

A few examples are given in the table below.

**Table 31-8:** Relationship Between CLKDIV and the Ratio of SCLK to EPPI Clock

| CLKDIV15-0 | SCLK:EPPI Clock Ratio |
|------------|-----------------------|
| DIV        |                       |
| 0x0002     | 1:3                   |
| 0x0003     | 1:4                   |
| 0x0002     | 1:3                   |
| 0x0003     | 1:4                   |
| ...        | ...                   |

## EPPI Operating Modes

The Enhanced Parallel Peripheral Interface (EPPI) supports various receive and transmit modes of operation which include the detection and generation of preamble data. Specifically, the EPPI supports data formats described in the specifications ITU-656, SMPTE 274M and SMPTE 296M. In addition to these modes, the EPPI also supports general purpose receive and transmit using up to three frame syncs (FS).

### ITU-R 656 Modes

The EPPI supports three input modes and one output mode for ITU-R 656 framed data. These modes are described in this section.

### ITU-R 656 Background

In ITU-R 656 (formerly known as CCIR-656) mode, the horizontal (H), vertical (V), and field (F) signals are sent as an embedded part of the video data stream in a series of bytes that form a control word.

The letter H is used to distinguish between the start of active video (SAV) and end of active video (EAV) signals, which indicate the beginning and end of active video data in each line. SAV occurs on a 1-to-0 transition of H, and EAV occurs on a 0-to-1 transition of H. The space between EAV and SAV is filled with horizontal blanking data. Therefore H = 1 during the horizontal blanking portion of the data stream and H = 0 during the active video portion of the data stream.

The letter V is used to denote the vertical blanking portion of the data stream. A transition in V can occur only in the EAV sequence. When  $V = 1$ , the data stream contains vertical blanking data and when  $V = 0$ , the data stream contains active video data.

The letter F is used to distinguish Field 1 from Field 2. Interlaced video has two fields in a frame of data. It requires each field to be handled uniquely, and alternate rows of each field combined to create the actual video image.

For interlaced video,  $F = 0$  represents Field 1 (“Odd Field”) and  $F = 1$  represents Field 2 (“Even Field”). Progressive video makes no distinction between Field 1 and Field 2, and F is always 0 for progressive video. Interlaced video requires each field to be handled uniquely, because alternate rows of each field combine to create the actual video image.

An entire field of video is comprised of active video plus horizontal blanking (the space between an EAV and SAV code) and vertical blanking (the space where  $V = 1$ ). A field of video commences on a transition of the F bit.



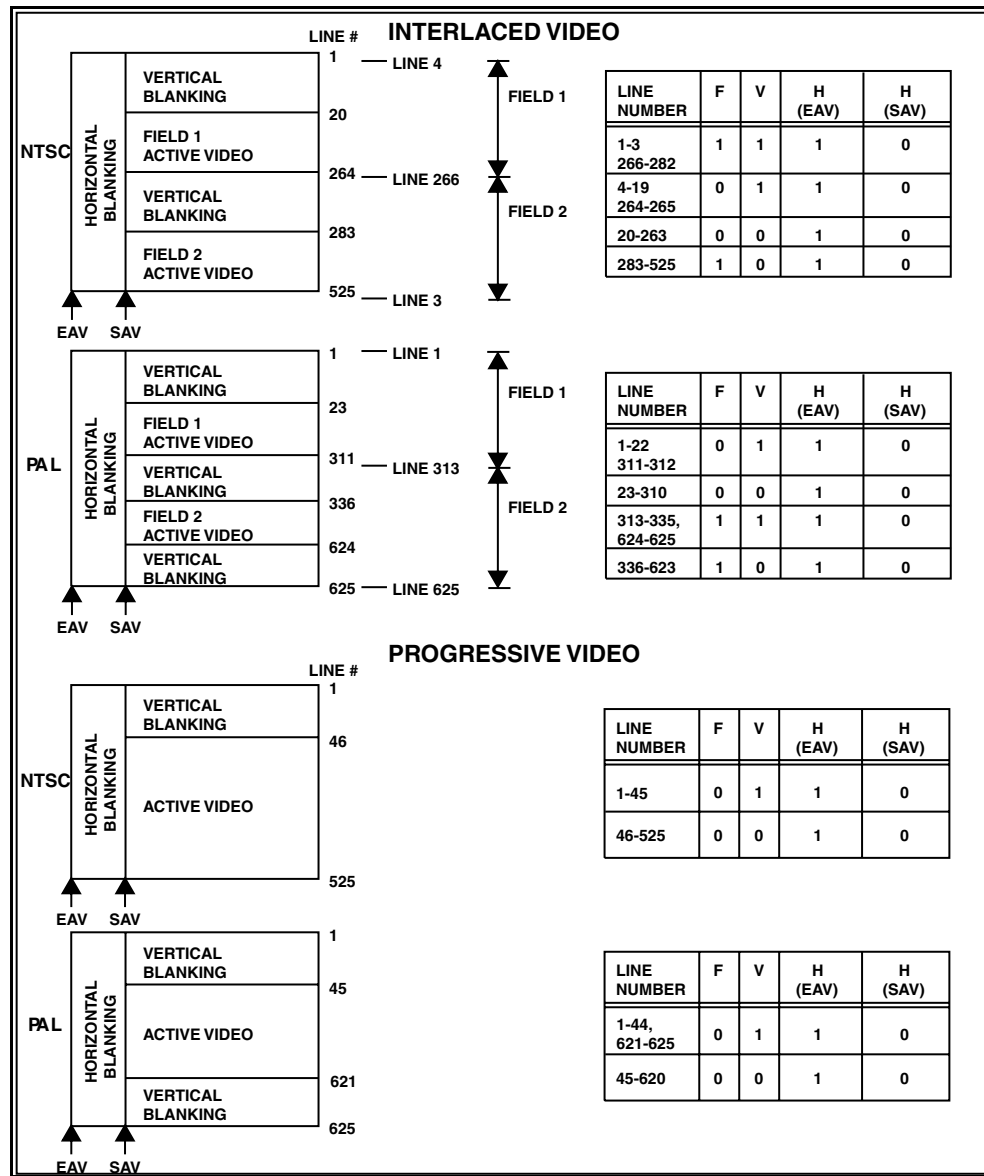


**Figure 31-9:** Typical Video Frame Partitioning for NTSC/PAL Systems in Interlaced and Progressive ITU-R BT.656 Systems

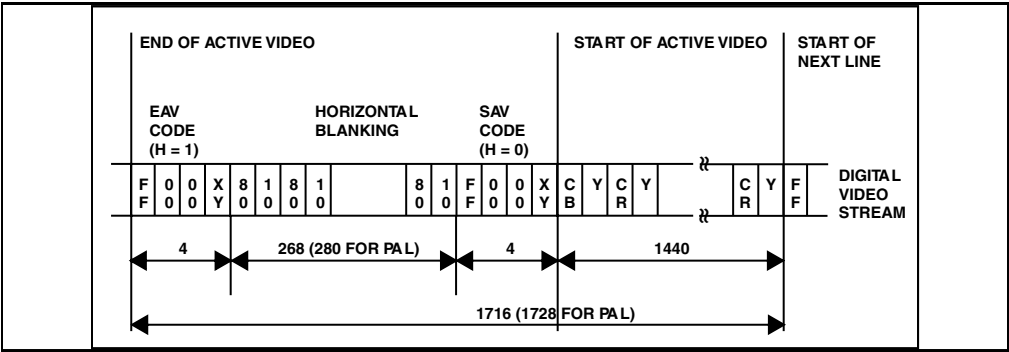


Figure 31-10: ITU-R 656 8-Bit Parallel Data Stream fro NTSC (PAL) Systems

Note there is a defined preamble of three data elements (for example, in the case of 8-bit video: 0xFF, 0x00, 0x00), followed by the XY status word, which, aside from the F (field), V (vertical blanking) and H (horizontal blanking) bits, contains four protection bits for error detection and correction. Note F and V are only allowed to change as part of EAV sequences (that is, transition from H = 0 to H = 1).

Table 31-9: Control Sequences for 8-Bit and 10-Bit ITU-R 656 Video

|              | 8-Bit Data |    |    |    |    |    |    |    | 10-Bit Data |    |
|--------------|------------|----|----|----|----|----|----|----|-------------|----|
|              | D9 (MSB)   | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1          | D0 |
| Preamble     | 1          | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1           | 1  |
|              | 0          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0           | 0  |
|              | 0          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0           | 0  |
| Control Byte | 1          | F  | V  | H  | P3 | P2 | P1 | P0 | 0           | 0  |

The bit definitions are as follows:

- F = 0 for field 1
- F = 1 for field 2
- V = 1 during vertical blanking
- V = 0 when not in vertical blanking
- H = 0 at SAV
- H = 1 at EAV
- $P3 = V \text{ XOR } H$
- $P2 = F \text{ XOR } H$

- $P1 = F \text{ XOR } V$
- $P0 = F \text{ XOR } V \text{ XOR } H$

P3–P0 are protection bits and enable 1 and 2-bit errors to be detected, and 1-bit errors to be corrected, at the receiver. The EPPI performs the correction if it detects 1-bit errors in F, V, or H. Errors in the protection bits themselves are detected but not corrected.

The EPPI\_STAT register contains two bits, ERRDET and ERRNCOR, used to report the status of error detected and error not corrected, respectively.

The ERRDET bit is set whenever an error is detected in the status word. However, this bit does not generate an interrupt. The ERRNCOR bit is set when more than a 1-bit error is detected in the status word. An interrupt is generated when the ERRNCOR bit is set. It can be cleared by clearing the ERRNCOR and ERRDET bits in the EPPI\_STAT register. Both bits are sticky and W1C.

In many applications, video streams other than the standard NTSC/PAL formats (for example, CIF, QCIF) can be employed. Because of this, the processor interface is flexible enough to accommodate different row and field lengths. In general, as long as the incoming video has the proper EAV/SAV codes, the EPPI can read it in. In other words, a CIF image could be formatted to be “656-compliant,” where EAV and SAV values define the range of the image for each line, and the V and F codes are used to delimit fields and frames.

The following sections provide descriptions of EPPI operations.

**Table 31-10:** Operating Modes and Generic EPPI Operation

|                 |               | How to configure                    | Useful for                                                             | How to configure in ITU R 656 TX Mode          |
|-----------------|---------------|-------------------------------------|------------------------------------------------------------------------|------------------------------------------------|
| ITU-R BT.656 RX | Entire field  | DIR = 0 XFRTYPE = b#01              |                                                                        |                                                |
|                 | Active video  | DIR = 0 XFRTYPE = b#00              |                                                                        |                                                |
|                 | Blanking only | DIR = 0 XFRTYPE = b#10              |                                                                        |                                                |
|                 | TX            | DIR = 1 XFRTYPE = b#11 FSCFG = b#00 | Applications where periodic frame syncs are not used to frame the data | BLANKGEN = 1<br>DLEN = (b#000, b#001 or b#100) |
|                 | RX            | DIR = 0 XFRTYPE = b#11 FSCFG = b#00 |                                                                        |                                                |

**Table 31-10:** Operating Modes and Generic EPPI Operation (Continued)

|         |    | How to configure                        | Useful for                                                                        | How to configure in ITU R 656 TX Mode          |
|---------|----|-----------------------------------------|-----------------------------------------------------------------------------------|------------------------------------------------|
| GP 1 FS | TX | DIR = 1 XFRTYPE = b#11 FSCFG = b#01     | Interfacing with ADCs, DACs and other general-purpose devices                     | BLANKGEN = 1<br>DLEN = (b#000, b#001 or b#100) |
|         | RX | DIR = 0 XFRTYPE = b#11 FSCFG = b#01     |                                                                                   |                                                |
| GP 2 FS | TX | DIR = 1 XFRTYPE = b#11 FSCFG = b#10     | Video applications that use two hardware synchronization signals, HSYNC and VSYNC | BLANKGEN = 1<br>DLEN = (b#000, b#001 or b#100) |
|         | RX | DIR = 0 XFRTYPE = b#11<br>FS_CFG = b#10 |                                                                                   |                                                |
| GP 3 FS | TX | DIR = 1 XFRTYPE = b#11 FSCFG = b#11     | Video applications that use three hardware sync signals, HSYNC, VSYNC, and FIELD  | BLANKGEN = 1<br>DLEN = (b#000, b#001 or b#100) |
|         | RX | DIR = 0 XFRTYPE = b#11 FSCFG = b#11     |                                                                                   |                                                |

## ITU-R 656 Input Modes

In the ITU-R 656 input modes, the clock CLK is either provided by the video source or supplied externally by the system.

There are three sub-modes supported for ITU-R 656 inputs: entire field, active video only, and vertical blanking interval only.

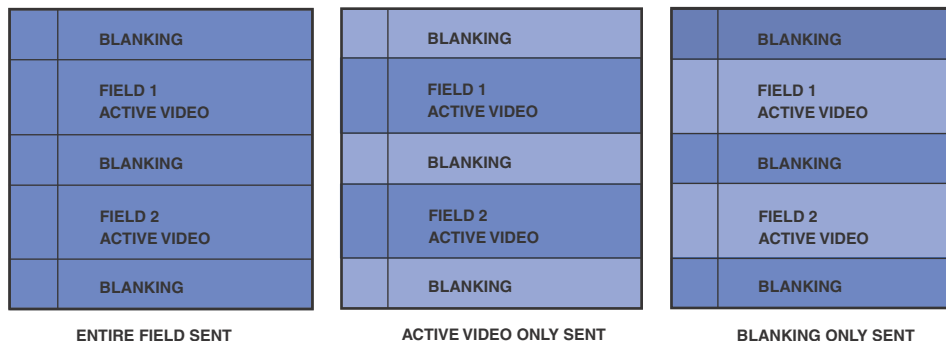


**Figure 31-11:** ITU-R 656 Input Sub-modes

## Entire Field

In this mode, the entire incoming bit stream is read in through the EPPI. This includes active video as well as control byte sequences and ancillary data that may be embedded in horizontal and vertical blanking intervals.

Data transfer starts immediately after synchronization to Field 1 occurs, but does not include the first EAV code that contains the  $F = 0$  assignment for interlaced video, or  $V = 0$  assignment for progressive video.

## Active Video

This mode is used when only the active video portion of a field is of interest, and not any of the blanking intervals. The EPPI ignores (does not read in) all data between EAV and SAV, as well as all data present when  $V = 1$ . In this mode, the control byte sequences are not stored to memory, they are filtered out by the EPPI. After synchronizing to the start of Field 1, the EPPI ignores incoming samples until it sees an SAV.

In active video mode, programs must specify the number of total (active plus vertical blanking) lines per frame in the `EPPI_FRAME` MMR, and the number of total (active plus horizontal blanking plus 8) samples per line in the `EPPI_LINE` MMR.

In this mode (as well as vertical blanking interval mode), any input data sequence that is considered as part of the preamble (for example in 8-bit ITU mode, 0xFF or 0x00 appearing in the input data stream are considered as part of the preamble) is not sent to memory, even if it appears individually, and is not tagged along with the preamble sequence FF, 00, 00.

## Vertical Blanking Interval (VBI)

In this mode, data transfer is only active while  $V = 1$  is in the control byte sequence. This indicates that the video source is in the midst of the Vertical Blanking Interval (VBI), which is sometimes used for ancillary data transmission. The ITU-R 656 recommendation specifies the format for these ancillary data packets, but the EPPI is not equipped to decode the packets themselves. This task must be handled in software. Horizontal blanking data is logged where it coincides with the rows of the VBI.

The VBI is split into two regions within each field. From the EPPI's standpoint, it considers these two separate regions as one contiguous space. However, keep in mind that frame synchronization begins at the start of Field 1, which doesn't necessarily correspond to the start of vertical blanking. For instance, in 525/60 systems, the start of Field 1 (F = 0) corresponds to line 4 of the VBI.

In VBI mode, the program must specify the number of total (active plus vertical blanking) lines per frame in the `EPPI_FRAME` MMR, and the number of total (active plus horizontal blanking plus 8) samples per line in the `EPPI_LINE` MMR.

In this mode (as well as active video mode), any input data sequence that is considered as part of the preamble (for example in 8 bit ITU mode, 0xFF or 0x00 appearing in the input data stream are considered as part of the preamble) is not sent to memory, even if it appears individually, and is not be tagged along with the preamble sequence FF, 00, 00.

## ITU-R 656 Output in GP Transmit Modes

In GP transmit mode, the EPPI provides the ability to frame an ITU-R 656 output stream with the proper preambles and blanking intervals. This is done by setting the `BLANKGEN` bit in the `EPPI_CTL` register. The EPPI then only needs to fetch active data from memory through the DMA channel, saving DMA bandwidth. The `EPPI_FS1_PASPL`, `EPPI_FS2_WLVB`, `EPPI_FS2_PALPF` and `EPPI_FS1_WLHB` registers need to be programmed correctly in order for the EPPI to internally generate and embed the proper preamble, status word (EAV and SAV sequences) and blanking data along with the active video from memory. The EPPI can also drive out the frame syncs based on the `FSCFG` setting in the `EPPI_CTL` register.

The figure below shows the bit stream format in 16-bit transmit modes with blanking generation (BLANKGEN enabled). Each 16-bit data sample consists of 8-bit luma (Y) and 8-bit chroma (Cr or Cb) components. During transmission, the chroma data and the 0x80 blanking bytes are on the upper half (MSBs) of the data lines while the luma data and the 0x10 blanking bytes are on the lower half (LSBs) of the data lines.

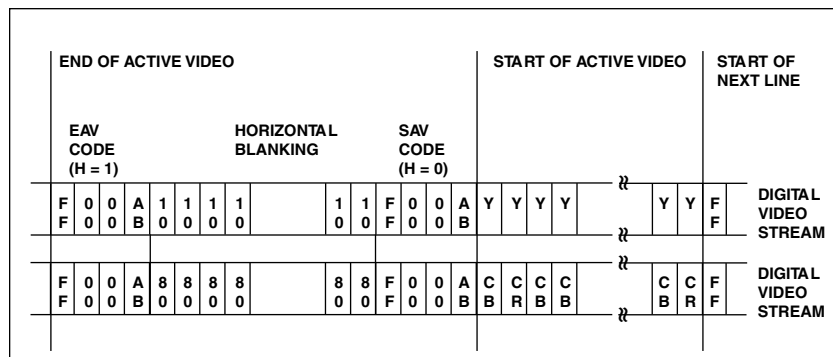


**Figure 31-12: 16-Bit Transmit with Internal Blanking Generation**

The following figure shows the data transmitted by the EPPI in this mode. After the EPPI is enabled and if the EPPI FIFO is not empty, the transmission starts by sending out a EAV sequence for a vertical blanking line. For interlaced video, F starts at 1. For progressive video, F is always 0.

**NOTE:** Internal blanking generation functionality is valid only when the data length is 8, 10, or 16 bits and when the EPPI is in GP transmit modes. The `BLANKGEN` bit generates preambles even in GP 2FS mode.

The ITU-R 656 Output mode's internal blanking generation functionality can also be bypassed (for instance, if sending ancillary data in the blanking interval) by clearing the `BLANKGEN` bit in the `EPPI_CTL` register. The `BLANKGEN` bit generates preambles even in GP 2FS mode.

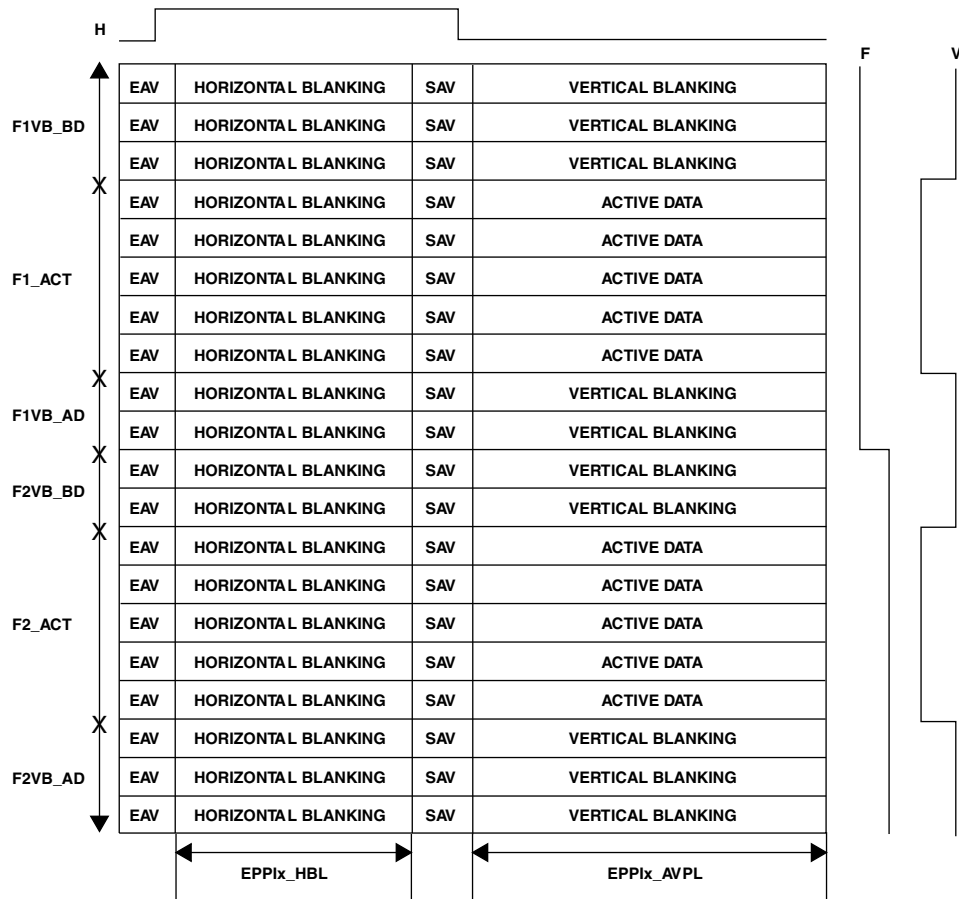


**Figure 31-13:** Generated Blanking Preamble Sequence

## Frame Synchronization in ITU-R 656 Modes

For interlaced video, the start of frame synchronization occurs when a high-to-low transition is detected in `F`, the field indicator. For progressive video, the start of frame synchronization occurs when a high-to-low transition is detected in `V`, the vertical blanking indicator. These transitions in `F` and `V` can occur only in the `EAV` sequence. A start of line is detected on a low-to-high transition in `H`, the horizontal blanking indicator, and this happens in the `EAV` sequence as well.

For interlaced video, the start of frame corresponds to the start of field 1. Consequently, up to two fields may be ignored (for example, if field 1 started before the EPPI-to-camera channel was established) before

data is received into the EPPI. For progressive video, the start of frame corresponds to the start of active video.

Because all H and V signaling is embedded in the data stream in ITU-R 656 modes, the count registers (EPPI\_HCNT, EPPI\_VCNT) are ignored. However, the EPPI\_FRAME register is still used to check for synchronization errors. Therefore, this MMR must be programmed with the number of lines expected in each frame of video. The EPPI keeps track of the number of EAV-to-SAV transitions that occur from the start of a frame until it decodes the end-of-frame condition (transition from F = 1 to F = 0 in the case of interlaced video and transition from V = 1 to V = 0 in the case of progressive video).

At this time, the actual number of lines processed is compared against the value in EPPI\_FRAME. If there is a mismatch, a frame track error is asserted in the EPPI\_STAT register. For instance, if an SAV transition was missed, the current field only has NUM\_ROWS – 1 rows, but re synchronization occurs at the start of the next frame. When the entire field is received, the field status bit is toggled in the EPPI\_STAT register. This way, an interrupt service routine (ISR) can discern which field was just read in.

## Ignoring Premature External Frame Syncs for Data Consistency

The relationship between the frame sync (FS) and the amount of data sent to memory is maintained even if frame syncs occur prematurely, whether due to data loss or an errant frame sync.

## General-Purpose EPPI Modes

The general-purpose (GP) EPPI modes are intended to suit a wide variety of data capture and transmission applications. Each EPPI has three bidirectional frame sync pins. Frame syncs can be generated internally by the EPPI, or by an external device communicating with the EPPI.

GP modes can be distinguished based on the number of frame syncs used. The EPPI supports the following GP modes:

- GP 0 FS mode
- GP 1 FS mode
- GP 2 FS mode
- GP 3 FS mode

All the GP modes, except 0 FS mode, support horizontal windowing. GP modes with 2 and 3 frame syncs also support vertical windowing.

For GP TX modes with internal clock or internal frame syncs, the EPPI starts generating the clock or frame syncs only when the EPPI FIFO is full for the first time. For GP 0 FS TX mode, the EPPI only starts transmitting when the EPPI FIFO is full for the first time.



## GP 0 FS Mode

This mode is useful for applications where periodic frame syncs are not used to frame the data.

GP 0 FS receive mode is further divided into two sub-modes; internal trigger ( $FLDSEL = 1$ ) and external trigger ( $FLDSEL = 0$ ), based on how data transmission/reception is initiated. GP 0 FS transmit mode is always internally triggered. All subsequent data manipulation is handled through DMA.

After initial trigger, the EPPI receives/transmits data samples on every clock cycle. However, if the `SKIPEN` bit is set for receive mode, the EPPI receives only alternate data samples.

The `EPPI_LINE`, `EPPI_FRAME`, `EPPI_HCNT`, `EPPI_HDLY`, `EPPI_VCNT` and `EPPI_VDLY` registers are not valid for GP 0 FS mode. Therefore windowing is not possible in this mode. Also, line and frame track errors are not applicable in this mode.

## Frame Synchronization in GP 0 FS External Trigger Mode

When the EPPI is programmed in external trigger mode, it does not generate the `EPPI_FS1` signal and a trigger must be provided by the external device. The EPPI starts receiving the data as soon as an `EPPI_FS1` signal assertion is detected. After that, all subsequent data manipulation is handled by way of DMA and any activity on `EPPI_FS1` is ignored.

## Frame Synchronization in GP 0 FS Internal Trigger Mode

When the EPPI is programmed in internal trigger mode, it starts receiving/transmitting data as soon as the EPPI clock is enabled and synchronized. When the EPPI is programmed in internal trigger mode, the EPPI starts receiving/transmitting data as soon as the EPPI clock is enabled and synchronized.

There may be up to two cycles latency before valid data is received or transmitted.

## GP 1 FS Mode

This mode is useful for interfacing the EPPI with analog-to-digital converters (ADCs), digital-to-analog converters (DACs) and other general- purpose devices. This mode works for both transmit and receive.

The `EPPI_FRAME`, `EPPI_VDLY` and `EPPI_VCNT` registers have no effect in GP 1 FS mode. As a result, frame track errors and vertical windowing are not possible in this mode.

## GP 2 FS Mode

GP 2 FS mode is useful for video applications that use two hardware synchronization signals, `HSYNC` and `VSYNC`. The `HSYNC` can be connected to `EPPI_FS1` and `VSYNC` can be connected to `EPPI_FS2`.

## DEN functionality in GP 2 FS Transmit Mode

When the EPPI is configured in GP 2 FS TX mode and when the EPPI is configured for internal frame sync generation, with EPPI\_CTL.MUXSEL and EPPI\_CTL.CLKGATEN bits no enabled, the EPPI\_FS3 pin functions as a data enable (DEN) pin. The functionality of the DEN pin is described in the following two cases.

Case 1 – When blanking generation (BLANKGEN) is enabled and the EPPI data length (DLEN) is configured for 8, 10, or 16-bit transfers, the EPPI\_FS3 pin asserts during the “active data” regions, aligned with EPPI\_CLK according to the clock polarity (POLC) settings. For this mode EPPI\_FS3 will be driven based on the POLC setting (i.e. EPPI\_FS3 will be driven out on the same EPPI clock edge that drives out data). The frame sync polarity (POLS) setting does not apply here— EPPI\_FS3 is always active high in this mode.

Case 2 – When blanking generation (BLANKGEN) is disabled, or it is enabled but the EPPI data length (DLEN) is configured for a transfer size different from 8, 10, or 16-bits, the EPPI\_FS3 pin asserts at the start of the active data region on each line, aligned with EPPI\_CLK according to the POLC settings. For this mode EPPI\_FS3 will be driven based on the POLC setting (i.e. EPPI\_FS3 will be driven out on the same EPPI clock edge that drives out data).

The POLS setting does not apply here -- EPPI\_FS3 is always active high in this mode. Once asserted, EPPI\_FS3 stays asserted for EPPI\_HCNT number of clock cycles per line, then it deasserts. This behavior on each line continues for the total number of lines programmed in the EPPI\_VCNT register per frame, and repeats at the start of subsequent video frames.

In case 2, if transmission of valid data is held off due to delays programmed in the EPPI\_HDLY and/or EPPI\_VDLY registers, the assertion of EPPI\_FS3 is also held off accordingly, on a per-line and/or per-frame basis.

## GP 3 FS Mode

This mode is useful for video applications that use three hardware synchronization signals, HSYNC, VSYNC, and FIELD. The HSYNC can be connected to the EPPI\_FS1 pin, VSYNC can be connected to the EPPI\_FS2 pin, and FIELD can be connected to the EPPI\_FS3 pin.

GP 3 FS mode is similar in operation to GP 2 FS mode, except that the start of frame synchronization in GP 3 FS mode also takes into account the EPPI\_FS3 pin. All the windowing register settings (EPPI\_FRAME, EPPI\_LINE, EPPI\_HDLY, EPPI\_HCNT, EPPI\_VDLY and EPPI\_VCNT), as well as data reception/transmission and error generation are the same as for GP 2 FS mode. In addition, for GP 3 FS mode with internal frame syncs, the FLDSEL bit setting in the EPPI\_CTL register specifies the condition under which the transfer should begin.

FS3 is generated by the EPPI and toggles during every assertion of FS2 or a combination of FS2 and FS1 (depending on the FLDSEL bit in the EPPI\_CTL register). The EPPI skips an FS2 if the FS3 value is high. Because of this condition, as a programming guideline, the FS2 period value should be half of the total number of pixels in the frame in 3 FS mode. When in 2FS mode, FS2 period should be programmed with the value equal to the number of pixels per frame.

## Supported Data Formats

The following sections describe EPPI receive and transmit data formats.

### EPPI Receive Data Format

**Table 31-11:** EPPI Receive Data Format

| Input Data Width | Usage Model   | Splitting/Packing Options                                                                                                                                                                                                                                                                                                        |
|------------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8                | NTSC/PAL data | SPLTEO = 1<br>SUBSPLTODD = 1 if required to separate chroma components                                                                                                                                                                                                                                                           |
|                  | RGB sensor    | No splitting possible.<br>PACKEN = 1; Four EPPI words are packed to 32bit DMA data.<br>PACKEN = 0; Each EPPI word is sent as an 8bit data on the 32bit DMA bus. This would consume 4 times the DMA bandwidth of the 8bit case with PACKEN = 1;                                                                                   |
|                  | ADCs          | Gives I (in phase) and Q (quadrature) components.<br>SPLTEO = 1<br>SUBSPLTODD = 0 since there are only two components.                                                                                                                                                                                                           |
| 10               | NTSC/PAL data | Each EPPI word is zero filled/sign extended to 16bits.<br>SPLTEO = 1.<br>SUBSPLTODD = 1 if required to separate chroma components.                                                                                                                                                                                               |
|                  | RGB sensor    | No splitting possible.<br>PACKEN = 1; Two EPPI words are zero filled/sign extended to 16 bits and packed to 32bit DMA data.<br>PACKEN = 0; Each EPPI word may be zero filled/sign extended to 16 bits and sent as a 16 bit data on the 32bit DMA bus. This would consume double the bandwidth of the 10bit case with PACKEN = 1; |
|                  | ADCs          | Each EPPI word is zero filled/sign extended to 16 bits.<br>SPLTEO = 1<br>SUBSPLTODD = 0 since there are only two components.                                                                                                                                                                                                     |

**Table 31-11:** EPPI Receive Data Format (Continued)

| Input Data Width | Usage Model                            | Splitting/Packing Options                                                                                                                                                                                                                                                                                                                  |
|------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12               | RGB sensor                             | No splitting possible.<br>PACKEN = 1; Two EPPI words are zero filled/sign extended to 16 bits and packed to 32bit DMA data.<br>PACKEN = 0; Each EPPI word may be zero filled/sign extended to 16 bits and sent as a 16 bit data on the 32bit DMA bus. This would consume double the bandwidth of the 12bit case with PACKEN = 1;           |
|                  | ADCs                                   | Each EPPI word is zero filled/sign extended to 16 bits.<br>SPLTEO = 1<br>SUBSPLTODD = 0 since there are only two components.                                                                                                                                                                                                               |
| 14               | ADCs                                   | Each EPPI word is zero filled/sign extended to 16 bits.<br>SPLTEO = 1<br>SUBSPLTODD = 0 since there are only two components.                                                                                                                                                                                                               |
| 16               | 8-bit luma/chroma pair for NTSC or HD  | SPLTEO = 1. SPLTWDRD = 1. SUBSPLTODD = 1 if required to separate chroma components                                                                                                                                                                                                                                                         |
|                  | 16-bit luma/chroma pair for NTSC or HD | SPLTEO = 1. SPLTWDRD = 0. SUBSPLTODD = 1 if required to separate chroma components                                                                                                                                                                                                                                                         |
|                  | RGB565 sensor                          | No splitting possible.<br>PACKEN = 1; Two EPPI words are packed to a 32 bit DMA data. RGBFMTEN is valid only in transmit modes. So, RGB565 cannot be made byte aligned in memory.<br>PACKEN = 0; Each EPPI word is sent as a 16 bit data on the 32 bit DMA bus. This would consume double the bandwidth of the 16 bit case with PACKEN = 1 |
|                  | 8-bit ADCs I/Q pair                    | SPLTEO = 1. SPLTWDRD = 1. SUBSPLTODD = 0.                                                                                                                                                                                                                                                                                                  |
| 20               | 16-bit ADCs I/Q pair                   | SPLTEO = 1. SPLTWDRD = 0. SUBSPLTODD = 0.                                                                                                                                                                                                                                                                                                  |
|                  | 10-bit luma/chroma pair for NTSC or HD | No support for 20 bit mode.                                                                                                                                                                                                                                                                                                                |

**Table 31-11:** EPPI Receive Data Format (Continued)

| Input Data Width | Usage Model   | Splitting/Packing Options                                                                                                                                                |
|------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24               | RGB888 sensor | No splitting possible.<br>PACKEN = 1; Four EPPI words are packed to three 32-bit DMA data.<br>PACKEN = 0; Each EPPI word is sent as a 24 bit data on the 32 bit DMA bus. |

## EPPI Transmit Data Format

**Table 31-12:** EPPI Transmit Data Format

| Output Data Width | Usage Model                          | Splitting/Packing Options                                                                                                                                                                                                                                                                                   |
|-------------------|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8                 | NTSC/PAL data                        | SPLTEO = 1<br>SUBSPLTODD = 1 if the chroma components (U and V) come in separate DMA words.                                                                                                                                                                                                                 |
|                   | serial RGB for lower-resolution LCDs | No splitting possible.<br>PACKEN = 1; The 32 bit DMA data will be unpacked to drive four EPPI words.<br>PACKEN = 0; The lowest 8 bits of the DMA data is driven on the EPPI data and the rest of the 24 bits are discarded. This would consume 4 times the DMA bandwidth of the 8 bit case with PACKEN = 1; |
| 10                | NTSC/PAL data                        | SPLTEO = 1<br>SUBSPLTODD = 1 if the chroma components (U and V) come in separate DMA words.                                                                                                                                                                                                                 |
|                   | DACs                                 | SPLTEO = 1. SUBSPLTODD = 0.                                                                                                                                                                                                                                                                                 |
| 12                | DACs                                 | SPLTEO = 1. SUBSPLTODD = 0.                                                                                                                                                                                                                                                                                 |
| 14                | DACs                                 | SPLTEO = 1. SUBSPLTODD = 0.                                                                                                                                                                                                                                                                                 |

**Table 31-12:** EPPI Transmit Data Format (Continued)

| Output Data Width | Usage Model                            | Splitting/Packing Options                                                                                                                                                                                                                            |
|-------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16                | 8-bit luma/chroma pair for NTSC or HD  | SPLTEO = 1. SPLTWRD = 1. SUBSPLTODD = 1 if the chroma components (U and V) come in separate DMA words.                                                                                                                                               |
|                   | 16-bit luma/chroma pair for NTSC or HD | SPLTEO = 1. SPLTWRD = 0. SUBSPLTODD = 1 if the chroma components (U and V) come in separate DMA words.                                                                                                                                               |
|                   | RGB565 LCD                             | No splitting possible.<br>RGBFMTEN = 1; Takes RGB888 data from the memory and drops the least significant bits from each component to drive out RGB565 data.                                                                                         |
|                   | 8-bit ADCs I/Q pair                    | SPLTEO = 1. SPLTWRD = 1. SUBSPLTODD = 0                                                                                                                                                                                                              |
|                   | 16-bit ADCs I/Q pair                   | SPLTEO = 1. SPLTWRD = 0. SUBSPLTODD = 1                                                                                                                                                                                                              |
| 18                | RGB666 LCD                             | No splitting possible.<br>RGBFMTEN = 1; Takes RGB888 data from the memory and drops the least significant 2 bits from each component to drive out RGB666 data.                                                                                       |
| 24                | RGB888 LCD                             | No splitting possible.<br>RGBFMTEN = 1; Takes three 32bit DMA data and drives out four 24bit data on the EPPI pins.<br>PACKEN = 0; Takes the 32bit DMA word, drops the most significant 8 bits and drives out the remaining 24 bits on the EPPI pins |

## EPPI Data Transfer Modes

The following sections describe EPPI data transfer modes, including receive/transmit data packing, sign extension, zero fill, receive/transmit split modes, clock gating, delayed start, and data consistency management.

### Data Packing for Receive Modes

For receive modes, if `EPPI_CTL.PACKEN = 1` and the DMA is a 32-bit DMA, the EPPI packs the incoming data into 32-bit words based on the `DLEN` and `SWAPEN` bit settings. When `SWAPEN = 0`, the EPPI puts the

first data in the least significant bits and when `SWAPEN = 1`, the EPPI puts the first data in the most significant bits. The packing options for the `DLEN` bits are as follows.

- When `DLEN = 8`, four 8-bit words can be packed into one 32-bit word.
- When `DLEN = 16`, two 16-bit words can be packed into one 32-bit word.
- For `DLEN` values that are more than 8 bits but less than 16 bits, two such words are either sign-extended or zero-filled to 16 bits, and packed into one 32-bit word.
- When `DLEN = 18`, the EPPI sign-extends or zero-fills the 18-bit data to 24 bits and packs four 24-bit words into three 32-bit words.
- When `DLEN = 24`, the EPPI packs four 24-bit words into three 32-bit words.

When `EPPI_CTL.PACKEN = 0`, the EPPI receives the incoming data and sends it on the bus as-is. If `DLEN` is less than or equal to 16 bits, the DMA is a 16-bit DMA; otherwise it is a 32-bit DMA.

## Data Packing for Transmit Modes

For transmit modes, if `EPPI_CTL.PACKEN = 1` and the DMA is a 32-bit DMA, the EPPI unpacks the 32-bit word according to the `DLEN` and `SWAPEN` bit settings.

If `SWAPEN = 1`, the EPPI transmits the most significant bits as the first data, and if `SWAPEN = 0`, the EPPI transmits the least significant bits as the first data. The unpacking options for the `DLEN` bits are as follows.

- When `DLEN = 8`, the EPPI transmits one 32-bit word from memory as four 8-bit data words.
- For `DLEN` values greater than 8 bits but less than or equal to 16 bits, the EPPI transmits one 32-bit word from memory as two 16-bit data words.
- When `DLEN = 18` or `DLEN = 24`, the EPPI transmits three 32-bit words from memory as four data words.

## Sign-Extension and Zero-Filling

The following list describes the bit settings and functionality for sign-extension and zero-fill.

- For `DLEN` equal to 10, 12 or 14, data is zero-filled or sign-extended to 16 bits.
- For `DLEN` equal to 18 bits, data is zero-filled or sign-extended to 24 bits if packing is enabled, and zero-filled or sign-extended to 32 bits if packing is disabled.
- For `DLEN` equal to 24 bits, data is zero-filled or sign-extended to 32 bits if packing is disabled.
- For `DLEN` equal to 8 bits, data is zero-filled or sign-extended to 16 bits if packing is disabled.
- If the `EPPI_CTL.SIGNEXT` bit is set (`SIGNEXT = 1`), then the data is sign-extended, otherwise it is zero-filled.

## Split Receive Modes

The EPPI\_CTL register has three control bits for Split Receive modes. These are SPLTE0, SUBSPLTODD and DMACFG. Packing is not valid in Split modes.

- If SPLTE0 is set, the EPPI splits the incoming data stream into two sub-streams, an even stream and an odd stream, and packs them separately.
- The SUBSPLTODD bit is available only when SPLTE0 is set. In this case, when SUBSPLTODD is set, the EPPI sub-splits the odd sub-stream, and packs the streams separately.
- The DMACFG bit is also available only if SPLTE0 is set. In this case if DMACFG is set, the EPPI uses two DMA channels and if DMACFG is cleared, the EPPI uses only one DMA channel.

## Split Transmit Modes

The EPPI\_CTL register has three control bits for Split Transmit modes. These are SPLTE0, SUBSPLTODD and DMACFG. The DMA is always a 32-bit DMA. Packing is not valid in Split modes.

- If SPLTE0 is set, the EPPI receives the Luma (Y3Y2Y1Y0) and interleaved Chroma (Cr1Cb1Cr0Cb0) data as 32 bits from the DMA channel and interleaves the data to form a 4:2:2 YCrCb data stream to transmit.
- The SUBSPLTODD bit is available only when SPLTE0 is set. In this case if SUBSPLTODD is set, the EPPI receives the Luma (Y3Y2Y1Y0) and de-interleaved Chroma (Cb3Cb2Cb1Cb0 and Cr3Cr2Cr1Cr0) and interleaves the data to form a 4:2:2 YCrCb data stream to transmit. (Note that the EPPI does not de-interleave the chroma data when formatting it into 4:2:2)
- DMACFG is also valid only if SPLTE0 is set. If DMACFG is set, the EPPI uses two DMA channels and if DMACFG is cleared, the EPPI uses only one DMA channel.

## Clock Gating

In ITU-R BT.656 and GP 0/1/2 FS modes, EPPI\_FS3 becomes a clock-gating input. This is valid for both internally and externally sourced EPPI\_CLK, in both RX and TX modes. This clock gating signal must be synchronous with EPPI\_CLK and must be driven by the external device on the rising edge of EPPI\_CLK. Its function is to hold the sync and data lines in their current state until EPPI\_FS3 is driven low. There are no additional latency cycles upon coming out of clock gating mode.

If clock gating is not required, the EPPI\_FS3 pin must either be tied to ground, or configured to operate as another of its multiplexed functions.

In GP 2 FS transmit mode with internally generated frame syncs, the EPPI\_FS3 pin functions as a data enable (DEN) pin.



## Support for Delayed Start of EPPI Frame Syncs

The EPPI supports a delayed start of the EPPI frame syncs—PPIx\_FS1 and PPIx\_FS2. The PPIx\_FS1D and PPIx\_FS2D registers are programmable registers corresponding to PPIx\_FS1 (HSYNC) and PPIx\_FS2 (VSYNC). These registers are described below.

The first active edge of the internally generated frame sync is delayed by the amount programmed in these registers starting from the first PPI\_CLK edge. The delay counter (which is the period counter itself, since they don't run together) runs only for the first time and then shuts off until the EPPI is re-enabled. The delay registers should be programmed prior to the first PPI\_CLK edge (similar to the width and period registers). The figure below shows the functioning of PPI\_FS1 and PPI\_FS2.

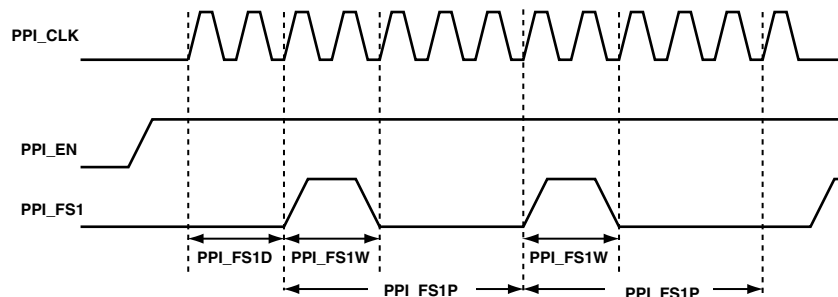


**Figure 31-14:** EPPI Delayed Frame Sync Generation

## Ignoring Premature External Frame Syncs for Data Consistency

Once a frame has started with a VSYNC followed by a HSYNC (or both coming together), a line is tracked. When the count expires, the state machine waits at the end of line for a HSYNC to come. With the arrival of the HSYNC, the state machine starts tracking the next line and so on.

The number of lines tracked is counted separately and once the end of a frame is reached, the state machine waits there for the next VSYNC/HSYNC combination and the next frame starts once they are sampled. The problem with this scheme is that every incoming FS (VSYNC or HSYNC) resets the respective counters and the tracking starts all over (even if the FS signals are premature). This results in incomplete data (or frames) to enter into memory through the PxP interface.

To correct this problem, the EPPI waits for a frame/line completion before considering any incoming FS as valid as described below.

- Single FS mode and line tracking in dual FS mode – When a line is in progress, if HSYNC is detected prematurely, it is ignored. A line track underflow event is generated.
- Dual FS mode – If a VSYNC is received when a frame is in progress, it is ignored. A Frame Track Underflow error (FTERRUNDR) is generated.

Ignoring the FS ensures that once a frame starts, the amount of data that goes into the memory/PxP interface exactly corresponds to the programmed data size in a frame.

**NOTE:** Even if the premature FS is a valid FS, the state machine loses at most one frame and it recovers in the subsequent FS. At all times the FS to number of data going into the memory relationship is maintained as programmed.

When data underflow errors occur at the DMA interface, the EPPI does the following.

- If a premature line sync is detected, a LT underflow error is generated. All further line track errors are ignored until a next valid line sync is detected by the EPPI.
- If a premature frame sync is detected, a FT underflow error is generated. All further frame track and line track errors are ignored until a next valid frame sync is detected by the EPPI.

## EPPI Event Control

The following sections describe how events are used in EPPI management.

## EPPI Status, Error and Interrupt Signals

The EPPI generates an error interrupt if any one of the following error conditions occur.

- YFIFOERR (YFIFO underflow or overflow)
- CFIFOERR (CFIFO underflow or overflow)
- LTERROVR (line track overflow error)
- LTERRUNDR (line track underflow error)
- FTERROVR (frame track overflow error)
- FTERRUNDR (frame track underflow error)
- ERRNCOR (ITU preamble error not corrected)

These conditions are flagged in the EPPI\_STAT register and cleared by a W1C (write-1-to-clear) operation. Each of the individual conditions which cause an EPPI error interrupt can be masked. The interrupt mask register (EPPI\_IMSK) allows the masking of individual conditions which cause error interrupts.

There is only one interrupt line from each EPPI so all interrupts are internally OR'ed and sent as a single interrupt to the core. The EPPI\_STAT register must then be read to discover specific errors. These errors are described in detail in the following sections.

## PxP Errors

The following are PxP generated errors.

Receive Mode – The receive data sent to the PxP is stalled by the destination in EPPI receive. In this situation the data is lost because the EPPI cannot hold off the incoming data. This condition is reported in the `PXPERR` bit in the `EPPI_STAT` register. Cleared by a `W1C`.

Transmit mode – The transmit data coming from the PxP interface goes through the FIFO. Data underflow errors are reported as `YFIFOERR` if the FIFO underflows similar to the DMA transmit mode.

## Frame and Line Track Errors

When functioning in external frame sync mode, the `LTERROVR` (line track error), `LERRUNDR` (line track error), `FTERROVR` (frame track error) and `FTERRUNDR` (frame track error) status bits are used to keep track of the line and frame synchronization errors. They are updated when there is a mismatch in the H sync and V sync detected by the EPPI as compared to the programmed values in `PPIX_LINE` and `PPIX_FRAME` count registers.

## LTERROVR and LERRUNDR Errors

The line track overflow (`LTERROVR`) and underflow errors (`LERRUNDR`) generate an interrupt (which can be masked) as soon as they are identified (not at the next FS). If the FS has not arrived when the `EPPI_LINE` counter expires, then a `LTERROVR` is generated. When the `EPPI_LINE` counter is running and an FS is detected, the `LERRUNDR` is generated. Both interrupts are cleared by a `W1C`.

## FTERROVR and FTERRUNDR Errors

In the EPPI, the frame track overflow (`FTERROVR`) and underflow errors (`FTERRUNDR`) generate an interrupt (can be masked) as soon as they are identified. At the instance when `EPPI_FRAME` counter expires, if the FS has not arrived, then a `FTERROVR` is reported. It doesn't wait for the next FS to report this. When the `EPPI_FRAME` counter is running, if an FS is detected, then `LERRUNDR` is reported. Both these generate an error interrupt and should be cleared by a `W1C` at their respective locations in the status register.

A premature frame sync will result in a frame track under run error but the error will be logged (register bit set) only after the subsequent blanking period (if any) present.

## Preamble Error Not Corrected Error

The EPPI supports data embedded frame syncs in ITU and SMPTE formats. In these formats, the PPI could receive an erroneous preamble which is not correctable. `ERRNCOR` signals when this event occurs.

## EPPI Programming Model

The following sections describe programming techniques, including receiving/transmitting ITU-R 656 frames; configuring transfers in GP0, GP1, GP2, and GP3 modes; and managing EPPI mode configurations.

## Receiving ITU-R 656 Frames

The EPPI supports the reception of ITU-R 656 compliant frames.

1. Configure EPPI to receive either full ITU-R 656 frame, active video or blanking information by setting the `XFRTYPE` bits accordingly in the `EPPI_CTL` register.
2. In both active video mode and in VBI (vertical blanking information) mode, specify the number of total (active plus vertical blanking) lines per frame in the `EPPI_FRAME` register, and the number of total (active plus horizontal blanking plus 8) samples per line in the `EPPI_LINE` register.
3. Configure DMA descriptors to move the data to memory.
4. Enable DMA.
5. Enable EPPI.

*RESULT:*

Depending on how the EPPI is configured, either the full ITU-R 656 frame will be moved to memory or just the active video or just the blanking information.

## Transmitting ITU-R 656 Frames in GP Transmit Modes

The EPPI can take active video from memory and generate the proper preambles and blanking information to produce valid ITU-R 656 video frames for transmission.

1. Provide active data frame in memory.
2. Set `BLANKGEN` bit in `EPPI_CTL` register to have EPPI generate blanking information.
3. Configure `EPPI_FS1_WLHB`, `EPPI_FS1_PASPL`, `EPPI_FS2_WLVB`, `EPPI_FS2_PALPF` registers accordingly.
4. Configure rest of EPPI settings.
5. Configure DMA to fetch active frame data from memory buffers.
6. Enable DMA
7. Enable EPPI

*RESULT:*

The EPPI will take the active data from memory and generate the blanking information and transmit an ITU-R 656 frame

## Configuring Transfers in GP 0 FS Mode

The EPPI can be configured so no periodic frame syncs are used to frame the data.

1. Configure the EPPI to operate in GP 0 FS mode by setting `XFRTYPE = b#11` and `FSCFG = b#00` in the `EPPI_CTL` register.
2. When receiving, configure the EPPI to trigger on internally or externally by setting the `FLDSEL` field appropriately in the `EPPI_CTL` register. When transmitting, the trigger will always be internally generated.
3. Configure DMA to move the data to or from memory.
4. Enable DMA.
5. Enable EPPI.

*RESULT:*

The data amount of data transferred will be controlled by the DMA descriptions and not by any frame syncs from the EPPI.

## Configuring Transfers in GP 1 FS Mode

GP 1 FS modes is useful for interfacing the EPPI with analog-to-digital converters (ADCs), digital-to-analog converters (DACs) and other general-purpose devices. This mode works for both transmit and receive.

**NOTE:** The `EPPI_FRAME`, `EPPI_VDLY` and `EPPI_VCNT` registers have no effect in GP 1 FS mode. As a result, frame track errors and vertical windowing are not possible in this mode.

1. Configure the EPPI to operate in GP 1 FS mode by setting `XFRTYPE = b#11` and `FSCFG = b#01` in the `EPPI_CTL` register. The frame syncs may be provided by an external device or they can be sourced by the EPPI itself.
2. Program the `EPPI_LINE` register to contain the number clock cycles expected between two assertions of `EPPI_FS1`. This is used to keep track of Line Track errors. It must be programmed before the `EPPI_HCNT` register.
3. Program the `EPPI_HDLY` register to contain the number of clock cycles to wait after the assertion of `EEPX_FS1`, for example, start of frame.
4. Program the `EPPI_HCNT` register to contain the number of data samples to receive or transmit for each frame.
5. Configure DMA to move the data to or from memory.

6. Enable DMA.
7. Enable EPPI.

*RESULT:*

Data will be moved in or out of memory and framed by a frame sync for every line.

## Configuring Transfers in GP 2 FS Mode

GP 2 FS mode is useful for video applications that use two hardware synchronization signals, HSYNC and VSYNC. The HSYNC can be connected to EPPI\_FS1 and VSYNC can be connected to EPPI\_FS2.

1. Configure the EPPI to operate in GP 0 FS mode by setting `XFRTYPE = b#11` and `FSCFG = b#10` in the `EPPI_CTL` register. The frame syncs may be provided by an external device or they can be sourced by the EPPI itself.
2. Program the `EPPI_FRAME` register to contain the number of expected lines per frame. It should be equal to the number of EPPI\_FS1 assertions expected between start of frame syncs and is used to keep track of frame track errors. It must be programmed before the `EPPI_VCNT` register.
3. Program the `EPPI_LINE` register to contain the number of clock cycles expected between two assertions of EPPI\_FS1. This is used to keep track of line track errors. It must be programmed before the `EPPI_HCNT` register.
4. Program the `EPPI_HDLY` register to contain the number of clock cycles to wait after the assertion of EPPI\_FS1, for example, start of line.
5. Program the `EPPI_HCNT` register to contain the number of data samples to receive or transmit for each line.
6. Program the `EPPI_VDLY` register to contain the number of lines to wait after the start of frame is detected.
7. Program the `EPPI_VCNT` register to contain the number of lines to receive or transmit.
8. If setting up the EPPI for transmit, the data enable (`DEN`) pin will behave according to the enabling of the blanking generation and setting of data length (`DLEN`). More detail can be found at TBD.
9. Enable DMA
10. Enable EPPI.

*RESULT:*

Data will be moved in or out of memory and framed by a frame sync for every line and frame.

## Configuring Transfers in GP 3 FS Mode

GP 3 FS mode is useful for video applications that use three hardware synchronization signals, HSYNC, VSYNC, and FIELD. The HSYNC can be connected to EPPI\_FS1, VSYNC can be connected to EPPI\_FS2, and FIELD can be connected to EPPI\_FS3.

1. Configure the EPPI to operate in GP 0 FS mode by setting the `XFRTYPE = b#11` and `FSCFG = b#11` in the `EPPI_CTL` register. The frame syncs may be provided by an external device or they can be sourced by the EPPI itself.
2. Configure the windowing registers according to steps in GP 2 FS Mode.
3. Enable DMA.
4. Enable EPPI.

### *RESULT:*

Data will be moved in or out of memory and framed by a frame sync for every line and frame. Operation and result is similar to operation in GP 2 FS Mode but EPPI\_FS3 will also be taken into account.

## Configuring the EPPI to Use the Windowing Feature

Windowing is a useful feature for applications where the region of interest is smaller than the active video stream (for example, sensor calibration, auto-focusing, and others). It can result in significant DMA bandwidth reduction. The EPPI supports windowing for GP Input modes.

1. Program the `EPPI_FRAME` register with the number of lines the frame contains.
2. Program the `EPPI_LINE` register with the number of samples per line in the frame.
3. Program the `EPPI_VDLY` register with the number of lines to wait after the start of a new frame before starting to read/transmit data.
4. Program the `EPPI_VCNT` register with the number of lines to read in or write out after `EPPI_VDLY` number of lines from the start of the frame.
5. Program the `EPPI_HDLY` register with the number of clock cycles to delay after the assertion of `EPPI_FS1` is detected for the start of a new line.
6. Program the `EPPI_HCNT` register with the number of samples to read in or write out after `EPPI_HDLY` number of cycles have expired since the assertion of `EPPI_FS1`.

## EPPI Mode Configuration

This section describes EPPI mode configurations, including support for all EPPI transmit and receive modes.

### Configuring 8-Bit Receive Mode

For 8-bit non-split receive mode, if `EPPI_CTL.PACKEN = 1`, the EPPI packs four bytes of incoming data into a 32-bit word. Alternate even or odd samples may be skipped based on the `SKIPEN` and `SKIEO` bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the `SWAPEN` bit setting.

**Table 31-13:** 8-Bit Receive Mode with Packing Enabled

| Pin Data (8 bits) | DMA DATA<br>SKIPEN=0<br>SKIEO=X<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=0<br>SKIEO=X<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIEO=1<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIEO=0<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIEO=1<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIEO=0<br>SWAPEN=1<br>SIGNEXT=X |
|-------------------|----------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------|
| 0x11              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0x22              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0x33              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0x44              | 0x4433 2211                                              | 0x1122 3344                                              |                                                          |                                                          |                                                          |                                                          |
| 0x55              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0x66              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0x77              |                                                          |                                                          | 0x7755 3311                                              |                                                          | 0x1133 5577                                              |                                                          |
| 0x88              | 0x8877 6655                                              | 0x5566 7788                                              |                                                          | 0x8866 4422                                              |                                                          | 0x2244 6688                                              |
| 0x99              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0xAA              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0xBB              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |
| 0xCC              | 0xCCBB<br>AA99                                           | 0x99AA<br>BBCC                                           |                                                          |                                                          |                                                          |                                                          |
| 0xDD              |                                                          |                                                          |                                                          |                                                          |                                                          |                                                          |



**Table 31-13:** 8-Bit Receive Mode with Packing Enabled (Continued)

| Pin Data (8 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=1<br>SIGNEXT=X |
|-------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0xEE              |                                                           |                                                           |                                                           |                                                           |                                                           |                                                           |
| 0xFF              |                                                           |                                                           | 0xFFDD<br>BB99                                            |                                                           | 0x99BB<br>DDFF                                            |                                                           |
| 0x00              | 0x00FF<br>EEDD                                            | 0xDDEE<br>EFF00                                           |                                                           | 0x00EE<br>CCAA                                            |                                                           | 0xAACC<br>EE00                                            |

If PACKEN = 0, the DMA is a 16-bit DMA and the EPPI either sign-extends or zero-fills the bytes of incoming data into a 16-bit word. The SWAPEN bit has no effect if PACKEN = 0.

**Table 31-14:** 8-Bit Receive Mode with Packing Disabled

| Pin Data (8 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=1 |
|-------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x44              | 0x0044                                                    | 0x0044                                                    | 0x0044                                                    |                                                           |
| 0x55              | 0x0055                                                    | 0x0055                                                    |                                                           | 0x0055                                                    |
| 0x66              | 0x0066                                                    | 0x0066                                                    | 0x0066                                                    |                                                           |
| 0x77              | 0x0077                                                    | 0x0077                                                    |                                                           | 0x0077                                                    |
| 0x88              | 0x0088                                                    | 0xFF88                                                    | 0x0088                                                    |                                                           |
| 0x99              | 0x0099                                                    | 0xFF99                                                    |                                                           | 0xFF99                                                    |
| 0xAA              | 0x00AA                                                    | 0xFFAA                                                    | 0x00AA                                                    |                                                           |
| 0xBB              | 0x00BB                                                    | 0xFFBB                                                    |                                                           | 0xFFBB                                                    |

## Configuring 10/12/14-Bit Receive Modes

For 10, 12, or 14-bit non-split receive modes, the EPPI first either zero-fills or sign-extends the incoming data (depending on the SIGNEXT bit) into a 16-bit word. If PACKEN = 1, the EPPI then packs two of these words into one 32-bit word. Alternate even or odd samples may be skipped based on the SKIPEN and

SKIPEN bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the SWAPEN bit setting.

**Table 31-15:** 10-Bit Receive Mode with Sign Extension, with Packing Enabled

| Pin Data (10 bits) | MSB | DMA DATA<br>SKIPEN=0<br>SKIPEN=X<br>SWAPEN=0<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=0<br>SKIPEN=X<br>SWAPEN=1<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEN=1<br>SWAPEN=0<br>SIGNEXT=1 |
|--------------------|-----|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x111              | 0   |                                                           |                                                           |                                                           |
| 0x222              | 1   | 0xFE22 0111                                               | 0x0111 FE22                                               |                                                           |
| 0x333              | 1   |                                                           |                                                           | 0xFF33 0111                                               |
| 0x044              | 0   | 0x0044 FF33                                               | 0xff33 0044                                               |                                                           |
| 0x155              | 0   |                                                           |                                                           |                                                           |
| 0x266              | 1   | 0xFE66 0155                                               | 0x0155 FE66                                               |                                                           |
| 0x377              | 1   |                                                           |                                                           | 0xFF77 0155                                               |
| 0x088              | 0   | 0x0088 FF77                                               | 0xFF77 0088                                               |                                                           |

**Table 31-16:** 10-Bit Receive Mode with Sign Extension, with Packing Enabled

| Pin Data (10 bits) | MSB | DMA DATA<br>SKIPEN=1<br>SKIPEN=0<br>SWAPEN=0<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEN=1<br>SWAPEN=1<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEN=0<br>SWAPEN=1<br>SIGNEXT=1 |
|--------------------|-----|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x111              | 0   |                                                           |                                                           |                                                           |
| 0x222              | 1   |                                                           |                                                           |                                                           |
| 0x333              | 1   |                                                           | 0x0011 FF33                                               |                                                           |
| 0x044              | 0   | 0x0044 FE22                                               |                                                           | 0xFE22 0044                                               |
| 0x155              | 0   |                                                           |                                                           |                                                           |
| 0x266              | 1   |                                                           |                                                           |                                                           |

**Table 31-16:** 10-Bit Receive Mode with Sign Extension, with Packing Enabled (Continued)

| Pin Data (10 bits) | MSB | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=0<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=1<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=1<br>SIGNEXT=1 |
|--------------------|-----|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x377              | 1   |                                                           | 0x0155 FF77                                               |                                                           |
| 0x088              | 0   | 0x0088 FE66                                               |                                                           | 0xFE66 0088                                               |

**Table 31-17:** 10-Bit Receive Mode, with Zero-Fill, with Packing Enabled

| Pin Data (10 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=0<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=1<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=0<br>SIGNEXT=0 | DMA DATA<br>SKIP_EN=1<br>SKIP_EO=0<br>SWAPEN=0<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=1<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=1<br>SIGNEXT=0 |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x111              |                                                           |                                                           |                                                           |                                                             |                                                           |                                                           |
| 0x222              | 0x0222 0111                                               | 0x0111 0222                                               |                                                           |                                                             |                                                           |                                                           |
| 0x333              |                                                           |                                                           | 0x0333 0111                                               |                                                             | 0x0011 0333                                               |                                                           |
| 0x044              | 0x0044 0333                                               | 0x0333 0044                                               |                                                           | 0x0044 0222                                                 |                                                           | 0x0222 0044                                               |
| 0x155              |                                                           |                                                           |                                                           |                                                             |                                                           |                                                           |
| 0x266              | 0x0266 0155                                               | 0x0155 0266                                               |                                                           |                                                             |                                                           |                                                           |
| 0x377              |                                                           |                                                           | 0x0377 0155                                               |                                                             | 0x0155 0377                                               |                                                           |
| 0x088              | 0x0088 0377                                               | 0x0377 0088                                               |                                                           | 0x0088 0266                                                 |                                                           | 0x0266 0088                                               |

The table shows a 10-bit receive mode example when `PACKEN = 0`:

**Table 31-18:** 10-bit Receive Mode with Packing Disabled

| Pin Data (10 bits) | MSB | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=0 |
|--------------------|-----|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x111              | 0   | 0x0111                                                    | 0x0111                                                    | 0x0111                                                    |                                                           |

**Table 31-18:** 10-bit Receive Mode with Packing Disabled (Continued)

| Pin Data (10 bits) | MSB | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=1 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=0 |
|--------------------|-----|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x222              | 1   | 0xFE22                                                    | 0x0222                                                    |                                                           | 0x0222                                                    |
| 0x333              | 1   | 0xFF33                                                    | 0x0333                                                    | 0xFF33                                                    |                                                           |
| 0x044              | 0   | 0x0044                                                    | 0x0444                                                    |                                                           | 0x0444                                                    |
| 0x155              | 0   | 0x0155                                                    | 0x0155                                                    | 0x0155                                                    |                                                           |
| 0x266              | 1   | 0xFE66                                                    | 0x0266                                                    |                                                           | 0x0266                                                    |
| 0x377              | 1   | 0xFF77                                                    | 0x0377                                                    | 0xFF77                                                    |                                                           |
| 0x088              | 0   | 0x0088                                                    | 0x0088                                                    |                                                           | 0x0088                                                    |

## Configuring 16-Bit Receive Mode

For 16-bit non-split receive mode, if `PACKEN = 1`, the EPPI packs two 16-bit incoming data into one 32-bit word. Alternate even or odd samples may be skipped based on the `SKIPEN` and `SKIPEO` bits. The first incoming data can be placed either in the least significant bit positions or in the most significant bit positions, based on the `SWAPEN` bit setting.

**Table 31-19:** 16-Bit Receive Mode with Packing Enabled

| Pin Data (16 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=1<br>SIGNEXT=X |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x1111             |                                                           |                                                           |                                                           |                                                           |                                                           |                                                           |
| 0x2222             | 0x2222 1111                                               | 0x1111 2222                                               |                                                           |                                                           |                                                           |                                                           |
| 0x3333             |                                                           |                                                           | 0x3333 1111                                               |                                                           | 0x1111 3333                                               |                                                           |
| 0x4444             | 0x4444 3333                                               | 0x3333 4444                                               |                                                           | 0x4444 2222                                               |                                                           | 0x2222 4444                                               |
| 0x5555             |                                                           |                                                           |                                                           |                                                           |                                                           |                                                           |
| 0x6666             | 0x6666 5555                                               | 0x5555 6666                                               |                                                           |                                                           |                                                           |                                                           |

**Table 31-19:** 16-Bit Receive Mode with Packing Enabled (Continued)

| Pin Data (16 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=0<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=1<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=1<br>SIGNEXT=X |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x7777             |                                                           |                                                           | 0x7777 5555                                               |                                                           | 0x5555 7777                                               |                                                           |
| 0x8888             | 0x8888 7777                                               | 0x7777 8888                                               |                                                           | 0x8888 6666                                               |                                                           | 0x6666 8888                                               |

**Table 31-20:** 16-bit Receive Mode with Packing Disabled

| Pin Data (16 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=X |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x1111             | 0x1111                                                    | 0x1111                                                    |                                                           |
| 0x2222             | 0x2222                                                    |                                                           | 0x2222                                                    |
| 0x3333             | 0x3333                                                    | 0x3333                                                    |                                                           |
| 0x4444             | 0x4444                                                    |                                                           | 0x4444                                                    |
| 0x5555             | 0x5555                                                    | 0x5555                                                    |                                                           |
| 0x6666             | 0x6666                                                    |                                                           | 0x6666                                                    |
| 0x7777             | 0x7777                                                    | 0x7777                                                    |                                                           |
| 0x8888             | 0x8888                                                    |                                                           | 0x8888                                                    |

## Configuring 18-Bit Receive Mode

For 18-bit non-split receive mode, if `PACKEN = 0`, the EPPI zero-fills or sign-extends the incoming data into a 32-bit word. If `PACKEN = 1`, the EPPI first zero-fills or sign-extends the incoming data to 24 bits, and then packs four such 24-bit data words into three 32-bit words. Alternate even or odd samples may be skipped based on the `SKIPEN` and `SKIPEO` bits. The `SWAPEN` bit has no effect.

**Table 31-21:** 18-bit Receive Mode with Packing Disabled

| Pin Data (18 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=0 |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x0 6666           | 0x0000 6666                                               | 0x0000 6666                                               |                                                           |
| 0x1 7777           | 0x0001 7777                                               |                                                           | 0x0001 7777                                               |
| 0x2 8888           | 0x0002 8888                                               | 0x0002 8888                                               |                                                           |
| 0x3 9999           | 0x0003 9999                                               |                                                           | 0x0003 9999                                               |

**Table 31-22:** 18-bit Receive Mode with Packing Enabled

| Pin Data (18 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=0 |
|--------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x0 1122           |                                                           |                                                           |                                                           |
| 0x1 3344           | 0x4400 1122                                               |                                                           |                                                           |
| 0x2 5566           | 0x5566 0133                                               | 0x6600 1122                                               |                                                           |
| 0x3 7788           | 0x0377 8802                                               |                                                           | 0x8801 3344                                               |
| 0x0 99AA           |                                                           | 0x99AA 0255                                               |                                                           |
| 0x1 BBCC           | 0xCC00 99AA                                               |                                                           | 0xBBCC 0377                                               |
| 0x2 DDEE           | 0xDDEE 01BB                                               | 0x02DD EE00                                               |                                                           |
| 0x3 FF12           | 0x03FF 122D                                               |                                                           | 0x03FF 1201                                               |

## Configuring 24-Bit Receive Mode

For 24-bit non-split receive mode, if `PACKEN = 0`, the EPPI zero-fills or sign-extends the incoming data into a 32-bit word. If `PACKEN = 1`, the EPPI packs four incoming 24-bit data words into three 32-bit words. Alternate even or odd samples may be skipped based on the `SKIPEN` and `SKIPEO` bits. The `SWAPEN` bit has no effect.

**Table 31-23:** 24-bit Receive Mode with Packing Disabled

| Pin Data(24 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X<br>SIGNEXT=0 | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X<br>SIGNEXT=0 |
|-------------------|-----------------------------------------------------------|-----------------------------------------------------------|-----------------------------------------------------------|
| 0x6 6666          | 0x0066 6666                                               | 0x0066 6666                                               |                                                           |
| 0x7 7777          | 0x0077 7777                                               |                                                           | 0x0077 7777                                               |
| 0x8 8888          | 0x0088 8888                                               | 0x0088 8888                                               |                                                           |
| 0x9 9999          | 0x0099 9999                                               |                                                           | 0x0099 9999                                               |

**Table 31-24:** 24-bit Receive Mode with Packing Enabled

| Pin Data (24 bits) | DMA DATA<br>SKIPEN=0<br>SKIPEO=X<br>SWAPEN=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=1<br>SWAPEN=X | DMA DATA<br>SKIPEN=1<br>SKIPEO=0<br>SWAPEN=X |
|--------------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|
| 0x11 2233          |                                              |                                              |                                              |
| 0x44 5566          | 0x6611 2233                                  |                                              |                                              |
| 0x77 8899          | 0x8899 4455                                  | 0x9911 2233                                  |                                              |
| 0x00 AABB          | 0x00AA BB77                                  |                                              | 0xBB44 5566                                  |
| 0xCC DDEE          |                                              | 0xDDEE 7788                                  |                                              |
| 0xFF 1234          | 0x34CC DDEE                                  |                                              | 0x1234 00AA                                  |
| 0x56 7890          | 0x7890 FF12                                  | 0x5678 90CC                                  |                                              |
| 0xAB CDEF          | 0xABCD EF56                                  |                                              | 0xABCD EFFF                                  |

## Configuring 8-Bit Split Receive Mode

For 8-bit split receive mode, `PACKEN` and `SIGNEXT` are not valid. The EPPI always packs four bytes of data into one 32-bit word.

**Table 31-25:** 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 0

| Pin<br>Data<br>(8 bits) | SPLTEO=1<br>SUBSPLTODD= 0<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             | SPLTEO=1<br>SUBSPLTODD= 1<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             |
|-------------------------|---------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|
|                         | DMACFG=1                                                      |                                                             | DMACFG=0                                                    | DMACFG=1                                                      |                                                             | DMACFG =0                                                   |
|                         | PRIMARY<br>DMA<br>CHANNEL                                     | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   | PRIMARY<br>DMA<br>CHANNEL                                     | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   |
| V <sub>0</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| Y <sub>0</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| U <sub>0</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| Y <sub>1</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| V <sub>1</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| Y <sub>2</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| U <sub>1</sub>          |                                                               | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> |                                                               |                                                             |                                                             |
| Y <sub>3</sub>          | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>   |                                                             | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>   |                                                             | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> |
| V <sub>2</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| Y <sub>4</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| U <sub>2</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| Y <sub>5</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| V <sub>3</sub>          |                                                               |                                                             |                                                             |                                                               | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> |
| Y <sub>6</sub>          |                                                               |                                                             |                                                             |                                                               |                                                             |                                                             |
| U <sub>3</sub>          |                                                               | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> |                                                               | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> |                                                             |
| Y <sub>7</sub>          | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub>   |                                                             | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub>   |                                                             | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> |



**Table 31-25:** 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 0 (Continued)

| Pin Data<br>(8 bits) | SPLTEO=1<br>SUBSPLTODD= 0<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                             |                           | SPLTEO=1<br>SUBSPLTODD= 1<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                             |                                                             |
|----------------------|---------------------------------------------------------------|-----------------------------|---------------------------|---------------------------------------------------------------|-----------------------------|-------------------------------------------------------------|
|                      | DMACFG=1                                                      |                             | DMACFG=0                  | DMACFG=1                                                      |                             | DMACFG =0                                                   |
|                      | PRIMARY<br>DMA<br>CHANNEL                                     | SECONDARY<br>DMA<br>CHANNEL | PRIMARY<br>DMA<br>CHANNEL | PRIMARY<br>DMA<br>CHANNEL                                     | SECONDARY<br>DMA<br>CHANNEL | PRIMARY<br>DMA<br>CHANNEL                                   |
| V <sub>4</sub>       |                                                               |                             |                           |                                                               |                             | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> |

**Table 31-26:** 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 1

| Pin Data<br>(8 bits) | SPLTEO=1<br>SUBSPLTODD=0<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             | SPLTEO=1<br>SUBSPLTODD=1<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                             |                                                             |
|----------------------|--------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------|-----------------------------|-------------------------------------------------------------|
|                      | DMACFG=1                                                     |                                                             | DMACFG=0                                                    | DMACFG=1                                                     |                             | DMACFG=0                                                    |
|                      | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL | PRIMARY<br>DMA<br>CHANNEL                                   |
| V <sub>0</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| Y <sub>0</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| U <sub>0</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| Y <sub>1</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| V <sub>1</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| Y <sub>2</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |
| U <sub>1</sub>       |                                                              | V <sub>0</sub> U <sub>0</sub> V <sub>1</sub> U <sub>1</sub> | V <sub>0</sub> U <sub>0</sub> V <sub>1</sub> U <sub>1</sub> |                                                              |                             |                                                             |
| Y <sub>3</sub>       | Y <sub>0</sub> Y <sub>1</sub> Y <sub>2</sub> Y <sub>3</sub>  |                                                             | Y <sub>0</sub> Y <sub>1</sub> Y <sub>2</sub> Y <sub>3</sub> | Y <sub>0</sub> Y <sub>1</sub> Y <sub>2</sub> Y <sub>3</sub>  |                             | Y <sub>0</sub> Y <sub>1</sub> Y <sub>2</sub> Y <sub>3</sub> |
| V <sub>2</sub>       |                                                              |                                                             |                                                             |                                                              |                             |                                                             |

**Table 31-26:** 8-bit Split Receive Mode with SKIPEN = 0 and SWAPEN = 1 (Continued)

| Pin Data<br>(8 bits) | SPLTEO=1<br>SUBSPLTODD=0<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             | SPLTEO=1<br>SUBSPLTODD=1<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             |
|----------------------|--------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|
|                      | DMACFG=1                                                     |                                                             | DMACFG=0                                                    | DMACFG=1                                                     |                                                             | DMACFG=0                                                    |
|                      | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   |
| Y <sub>4</sub>       |                                                              |                                                             |                                                             |                                                              |                                                             |                                                             |
| U <sub>2</sub>       |                                                              |                                                             |                                                             |                                                              |                                                             |                                                             |
| Y <sub>5</sub>       |                                                              |                                                             |                                                             |                                                              |                                                             |                                                             |
| V <sub>3</sub>       |                                                              |                                                             |                                                             |                                                              | V <sub>0</sub> V <sub>1</sub> V <sub>2</sub> V <sub>3</sub> | V <sub>0</sub> V <sub>1</sub> V <sub>2</sub> V <sub>3</sub> |
| Y <sub>6</sub>       |                                                              |                                                             |                                                             |                                                              |                                                             |                                                             |
| U <sub>3</sub>       |                                                              | V <sub>2</sub> U <sub>2</sub> V <sub>3</sub> U <sub>3</sub> | V <sub>2</sub> U <sub>2</sub> V <sub>3</sub> U <sub>3</sub> |                                                              | U <sub>0</sub> U <sub>1</sub> U <sub>2</sub> U <sub>3</sub> |                                                             |
| Y <sub>7</sub>       | Y <sub>4</sub> Y <sub>5</sub> Y <sub>6</sub> Y <sub>7</sub>  |                                                             | Y <sub>4</sub> Y <sub>5</sub> Y <sub>6</sub> Y <sub>7</sub> | Y <sub>4</sub> Y <sub>5</sub> Y <sub>6</sub> Y <sub>7</sub>  |                                                             | Y <sub>4</sub> Y <sub>5</sub> Y <sub>6</sub> Y <sub>7</sub> |
| V <sub>4</sub>       |                                                              |                                                             |                                                             |                                                              |                                                             | U <sub>0</sub> U <sub>1</sub> U <sub>2</sub> U <sub>3</sub> |

For the case when SPLTEO = 1, SUBSPLTODD = 1 and DMACFG = 0, note that although the second Chroma component (U<sub>0</sub>U<sub>1</sub>U<sub>2</sub>U<sub>3</sub> in the tables above) sent over the DMA bus is completely packed before the Luma component (Y<sub>4</sub>Y<sub>5</sub>Y<sub>6</sub>Y<sub>7</sub> in the tables above), it is intentionally held until that previous word is moved out. This allows the separation of Luma and Chroma values into individual buffers when using 2D-DMA.

### Configuring 10/12/14/16-Bit Split Receive Mode with SPLTWRD=0

For 16-bit split receive mode, PACKEN is not valid. The EPPI always packs two 16-bit words into one 32-bit word. For 10, 12, or 14-bit split receive modes, the EPPI first either sign-extends or zero-fills the incoming data into a 16 bit word, and then packs two of these words into one 32-bit word to send to the DMA.

**Table 31-27:** 16-bit Split Receive Mode with SPLTWRD = 0, SKIPEN = 0 and SWAPEN = 0

| Pin Data<br>(16 bits) | SPLTEO=1<br>SUBSPLTODD=0<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                               |                               | SPLTEO=1<br>SUBSPLTODD=1<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                               |                               |
|-----------------------|--------------------------------------------------------------|-------------------------------|-------------------------------|--------------------------------------------------------------|-------------------------------|-------------------------------|
|                       | DMACFG=1                                                     |                               | DMACFG=0                      | DMACFG=1                                                     |                               | DMACFG=0                      |
|                       | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL   | PRIMARY<br>DMA<br>CHANNEL     | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL   | PRIMARY<br>DMA<br>CHANNEL     |
| V <sub>0</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| Y <sub>0</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| U <sub>0</sub>        |                                                              | U <sub>0</sub> V <sub>0</sub> | U <sub>0</sub> V <sub>0</sub> |                                                              |                               |                               |
| Y <sub>1</sub>        | Y <sub>1</sub> Y <sub>0</sub>                                |                               | Y <sub>1</sub> Y <sub>0</sub> | Y <sub>1</sub> Y <sub>0</sub>                                |                               | Y <sub>1</sub> Y <sub>0</sub> |
| V <sub>1</sub>        |                                                              |                               |                               |                                                              | V <sub>1</sub> V <sub>0</sub> | V <sub>1</sub> V <sub>0</sub> |
| Y <sub>2</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| U <sub>1</sub>        |                                                              | U <sub>1</sub> V <sub>1</sub> | U <sub>1</sub> V <sub>1</sub> |                                                              | U <sub>1</sub> U <sub>0</sub> |                               |
| Y <sub>3</sub>        | Y <sub>3</sub> Y <sub>2</sub>                                |                               | Y <sub>3</sub> Y <sub>2</sub> | Y <sub>3</sub> Y <sub>2</sub>                                |                               | Y <sub>3</sub> Y <sub>2</sub> |
| V <sub>2</sub>        |                                                              |                               |                               |                                                              |                               | U <sub>1</sub> U <sub>0</sub> |

**Table 31-28:** 16-bit Split Receive Mode with SPLTWRD = 0, SKIPEN = 0 and SWAPEN = 1

| Pin Data<br>(16 bits) | SPLTEO=1<br>SUBSPLTODD=0<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                               |                               | SPLTEO=1<br>SUBSPLTODD=1<br>SWAPEN=1<br>SKIPEN=0<br>SKIPEO=X |                               |                               |
|-----------------------|--------------------------------------------------------------|-------------------------------|-------------------------------|--------------------------------------------------------------|-------------------------------|-------------------------------|
|                       | DMACFG=1                                                     |                               | DMACFG=0                      | DMACFG=1                                                     |                               | DMACFG=0                      |
|                       | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL   | PRIMARY<br>DMA<br>CHANNEL     | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL   | PRIMARY<br>DMA<br>CHANNEL     |
| V <sub>0</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| Y <sub>0</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| U <sub>0</sub>        |                                                              | V <sub>0</sub> U <sub>0</sub> | V <sub>0</sub> U <sub>0</sub> |                                                              |                               |                               |
| Y <sub>1</sub>        | Y <sub>0</sub> Y <sub>1</sub>                                |                               | Y <sub>0</sub> Y <sub>1</sub> | Y <sub>0</sub> Y <sub>1</sub>                                |                               | Y <sub>0</sub> Y <sub>1</sub> |
| V <sub>1</sub>        |                                                              |                               |                               |                                                              | V <sub>0</sub> V <sub>1</sub> | V <sub>0</sub> V <sub>1</sub> |
| Y <sub>2</sub>        |                                                              |                               |                               |                                                              |                               |                               |
| U <sub>1</sub>        |                                                              | V <sub>1</sub> U <sub>1</sub> | V <sub>1</sub> U <sub>1</sub> |                                                              | U <sub>0</sub> U <sub>1</sub> |                               |
| Y <sub>3</sub>        | Y <sub>2</sub> Y <sub>3</sub>                                |                               | Y <sub>2</sub> Y <sub>3</sub> | Y <sub>2</sub> Y <sub>3</sub>                                |                               | Y <sub>2</sub> Y <sub>3</sub> |
| V <sub>2</sub>        |                                                              |                               |                               |                                                              |                               | U <sub>0</sub> U <sub>1</sub> |

### Configuring 16-Bit Split Receive Mode with SPLTWRD=1

For 16-bit split receive mode, PACKEN is not valid. The EPPI always packs two 16-bit words into one 32-bit word. The SPLTWRD bit is only valid when DLEN = 16 bits.

**Table 31-29:** 16-bit Split Receive Mode with SPLTWRD = 1, SKIPEN = 0 and SWAPEN = 0

| Pin Data<br>(16 bits)         | SPLTEO=1<br>SUBSPLTODD=0<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             | SPLT_EVEN_ODD=1<br>SUBSPLTODD=1<br>SWAPEN=0<br>SKIPEN=0<br>SKIPEO=X |                                                             |                                                             |
|-------------------------------|--------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|
|                               | DMACFG=1                                                     |                                                             | DMACFG=0                                                    | DMACFG=1                                                            |                                                             | DMACFG=0                                                    |
|                               | PRIMARY<br>DMA<br>CHANNEL                                    | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   | PRIMARY<br>DMA<br>CHANNEL                                           | SECONDARY<br>DMA<br>CHANNEL                                 | PRIMARY<br>DMA<br>CHANNEL                                   |
| V <sub>0</sub> Y <sub>0</sub> |                                                              |                                                             |                                                             |                                                                     |                                                             |                                                             |
| U <sub>0</sub> Y <sub>1</sub> |                                                              |                                                             |                                                             |                                                                     |                                                             |                                                             |
| V <sub>1</sub> Y <sub>2</sub> |                                                              |                                                             |                                                             |                                                                     |                                                             |                                                             |
| U <sub>1</sub> Y <sub>3</sub> | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>  | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub>         |                                                             | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> |
| V <sub>2</sub> Y <sub>4</sub> |                                                              |                                                             | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> |                                                                     |                                                             |                                                             |
| U <sub>2</sub> Y <sub>5</sub> |                                                              |                                                             |                                                             |                                                                     |                                                             |                                                             |
| V <sub>3</sub> Y <sub>6</sub> |                                                              |                                                             |                                                             |                                                                     | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> |
| U <sub>3</sub> Y <sub>7</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub>  | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub>         | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> |
| V <sub>4</sub> Y <sub>8</sub> |                                                              |                                                             | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> |                                                                     |                                                             | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> |

## Configuring 8-Bit Transmit Mode

For 8-bit non-split transmit mode, if PACKEN = 1, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory into four bytes to transmit. The EPPI transmits either the MSBs or the LSBs as the first data, depending on the SWAPEN bit setting. If PACKEN = 0, the DMA is a 16-bit DMA and the EPPI transmits the lower 8 bits. SWAPEN has no effect when PACKEN = 0.

**Table 31-30:** 8-bit Transmit Mode with Packing Enabled

| DMA Data (32 bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
| 0x11223344         | 0x44                   | 0x11                   |

**Table 31-30:** 8-bit Transmit Mode with Packing Enabled (Continued)

| DMA Data (32 bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
| 0x55667788         |                        | 0x33<br>0x22           |
|                    | 0x22                   | 0x33                   |
|                    | 0x11                   | 0x44                   |
|                    | 0x88                   | 0x55                   |
|                    | 0x77                   | 0x66                   |
|                    | 0x66                   | 0x77                   |
|                    | 0x55                   | 0x88                   |

**Table 31-31:** Data Sent in 8-bit Transmit Mode with Packing Disabled

| DMA Data (16 bits) | Pin Data SWAPEN=X |
|--------------------|-------------------|
| 0x1234             | 0x34              |
| 0x2345             | 0x45              |
| 0x3456             | 0x56              |

## Configuring 10/12/14-Bit Transmit Modes

For 10, 12, or 14-bit non-split transmit modes, if `PACKEN = 1`, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory into two 16-bit data words, then transmits the required LSBs from each. The EPPI transmits either the most significant word or the least significant word as the first data, depending on the `SWAPEN` bit setting. If `PACKEN = 0`, the DMA is a 16-bit DMA and the EPPI transmits the required LSBs. The `SWAPEN` bit has no effect when `PACKEN = 0`.

**Table 31-32:** 10-bit Transmit Mode with Packing Enabled

| DMA Data (32 bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
| 0x1111 2222        | 0x222                  | 0x111                  |
| 0x3333 4444        | 0x111                  | 0x222                  |
|                    | 0x044                  | 0x333                  |

**Table 31-32:** 10-bit Transmit Mode with Packing Enabled (Continued)

| DMA Data (32 bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
|                    | 0x333                  | 0x044                  |

**Table 31-33:** 10-bit Transmit Mode with Packing Disabled

| DMA Data (16 bits) | Pin Data SWAPEN=X |
|--------------------|-------------------|
| 0x1234             | 0x234             |
| 0x2345             | 0x345             |
| 0x3456             | 0x056             |
| 0x4567             | 0x167             |

## Configuring 16-Bit Transmit Mode

For 16-bit non-split transmit mode, if `PACKEN = 1`, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory into two 16-bit data words to transmit. The EPPI transmits either the MSBs or the LSBs as the first data, depending on the `SWAPEN` bit setting. If `PACKEN = 0`, the DMA is a 16-bit DMA and the EPPI transmits the data as is. `SWAPEN` has no effect when `PACKEN = 0`.

**Table 31-34:** 16-bit Transmit Mode with Packing Enabled

| DMA Data (32 bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
| 0x1111 2222        | 0x2222                 | 0x1111                 |
| 0x3333 4444        | 0x1111                 | 0x2222                 |
|                    | 0x4444                 | 0x3333                 |
|                    | 0x3333                 | 0x4444                 |

**Table 31-35:** 16-bit Transmit Mode with Packing Disabled

| DMA Data (16 bits) | Pin Data SWAPEN=X |
|--------------------|-------------------|
| 0x1234             | 0x1234            |
| 0x2345             | 0x2345            |

**Table 31-35:** 16-bit Transmit Mode with Packing Disabled (Continued)

| DMA Data (16 bits) | Pin Data SWAPEN=X |
|--------------------|-------------------|
| 0x3456             | 0x3456            |

## Configuring 18-Bit Transmit Mode

For 18-bit transmit mode, if `PACKEN = 1`, the DMA is a 32-bit DMA and the EPPI unpacks the 32-bit word from memory. In this mode, when `RGB_FMT_EN` is set, the least significant two bits of R, G, and B are dropped.

**Table 31-36:** 18-bit Transmit Mode with Packing Enabled

| DMA Data    | Pin Data (18-bits) |            |
|-------------|--------------------|------------|
|             | RGBFMTEN=0         | RGBFMTEN=1 |
| 0x0123 4567 | 0x3 4567           | 0x0 8459   |
| 0x89AB CDEF | 0x1 EF01           | 0x3 3EC0   |
| 0x0123 4567 | 0x3 89AB           | 0x1 98AA   |
|             | 0x1 2345           | 0x0 0211   |

**Table 31-37:** 18-bit Transmit Mode with Packing Disabled

| DMA Data    | Pin Data (18-bits) |            |
|-------------|--------------------|------------|
|             | RGBFMTEN=0         | RGBFMTEN=1 |
| 0x0123 4567 | 0x3 4567           | 0x0 8459   |
| 0x89AB CDEF | 0x3 CDEF           | 0x2 ACFB   |
| 0x0123 4567 | 0x3 4567           | 0x0 8459   |

## Configuring 24-Bit Transmit Mode

For 24-bit transmit mode, if `PACKEN = 1`, the DMA is a 32-bit DMA and the EPPI unpacks three 32-bit words from memory into four 24-bit words to transmit. The effect of the `SWAPEN` bit setting is shown in the table below.



**Table 31-38:** 24-bit Transmit Mode

| DMA Data (32 Bits) | Pin Data when SWAPEN=0 | Pin Data when SWAPEN=1 |
|--------------------|------------------------|------------------------|
| $R_1B_0G_0R_0$     | $B_0G_0R_0$            | $R_0G_0B_0$            |
| $G_2R_2B_1G_1$     | $B_1G_1R_1$            | $R_1G_1B_1$            |
| $B_3G_3R_3B_2$     | $B_2G_2R_2$            | $R_2G_2B_2$            |
|                    | $B_3G_3R_3$            | $R_3G_3B_3$            |

## Configuring 8-Bit Split Transmit Mode

For 8-bit split transmit mode, `PACKEN` is not valid. The EPPI always unpacks the 32-bit DMA data into four bytes to transmit.

**Table 31-39:** 8-bit Split Transmit Mode with `SPLTEO=1`, `SUBSPLTODD=0` and `SWAPEN=0`

| DMACFG=1<br>DMACFG=0 |                     |                   |                     |                   |
|----------------------|---------------------|-------------------|---------------------|-------------------|
| DMA0 DATA (32 bits)  | DMA1 DATA (32 bits) | Pin Data (8 bits) | DMA0 DATA (32 bits) | Pin Data (8 bits) |
| $Y_3Y_2Y_1Y_0$       | $U_1V_1U_0V_0$      | $V_0$             | $U_1V_1U_0V_0$      | $V_0$             |
| $Y_7Y_6Y_5Y_4$       | $U_3V_3U_2V_2$      | $Y_0$             | $Y_3Y_2Y_1Y_0$      | $Y_0$             |
|                      |                     | $U_0$             | $U_3V_3U_2V_2$      | $U_0$             |
|                      |                     | $Y_1$             | $Y_7Y_6Y_5Y_4$      | $Y_1$             |
|                      |                     | $V_1$             |                     | $V_1$             |
|                      |                     | $Y_2$             |                     | $Y_2$             |
|                      |                     | $U_1$             |                     | $U_1$             |
|                      |                     | $Y_3$             |                     | $Y_3$             |
|                      |                     | $V_2$             |                     | $V_2$             |
|                      |                     | $Y_4$             |                     | $Y_4$             |

**Table 31-39:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=0 and SWAPEN=0 (Continued)

| DMACFG=1<br>DMACFG=0 |                     |                   |                     |                   |
|----------------------|---------------------|-------------------|---------------------|-------------------|
| DMA0 DATA (32 bits)  | DMA1 DATA (32 bits) | Pin Data (8 bits) | DMA0 DATA (32 bits) | Pin Data (8 bits) |
|                      |                     | U <sub>2</sub>    |                     | U <sub>2</sub>    |
|                      |                     | Y <sub>5</sub>    |                     | Y <sub>5</sub>    |
|                      |                     | V <sub>3</sub>    |                     | V <sub>3</sub>    |
|                      |                     | Y <sub>6</sub>    |                     | Y <sub>6</sub>    |
|                      |                     | U <sub>3</sub>    |                     | U <sub>3</sub>    |
|                      |                     | Y <sub>7</sub>    |                     | Y <sub>7</sub>    |

**Table 31-40:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1 and SWAPEN=0

| DMACFG=1<br>DMACFG=0                                        |                                                             |                   |                                                             |                   |
|-------------------------------------------------------------|-------------------------------------------------------------|-------------------|-------------------------------------------------------------|-------------------|
| DMA0 DATA (32 bits)                                         | DMA1 DATA (32 bits)                                         | Pin Data (8 bits) | DMA0 DATA (32 bits)                                         | Pin Data (8 bits) |
| Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>0</sub>    | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>0</sub>    |
| Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | Y <sub>0</sub>    | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | Y <sub>0</sub>    |
|                                                             | V <sub>7</sub> V <sub>6</sub> V <sub>5</sub> V <sub>4</sub> | U <sub>0</sub>    | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | U <sub>0</sub>    |
|                                                             | U <sub>7</sub> U <sub>6</sub> U <sub>5</sub> U <sub>4</sub> | Y <sub>1</sub>    | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | Y <sub>1</sub>    |
|                                                             |                                                             | V <sub>1</sub>    |                                                             | V <sub>1</sub>    |
|                                                             |                                                             | Y <sub>2</sub>    |                                                             | Y <sub>2</sub>    |
|                                                             |                                                             | U <sub>1</sub>    |                                                             | U <sub>1</sub>    |
|                                                             |                                                             | Y <sub>3</sub>    |                                                             | Y <sub>3</sub>    |
|                                                             |                                                             | V <sub>2</sub>    |                                                             | V <sub>2</sub>    |
|                                                             |                                                             | Y <sub>4</sub>    |                                                             | Y <sub>4</sub>    |

**Table 31-40:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1 and SWAPEN=0 (Continued)

| DMACFG=1<br>DMACFG=0 |                     |                   |                     |                   |
|----------------------|---------------------|-------------------|---------------------|-------------------|
| DMA0 DATA (32 bits)  | DMA1 DATA (32 bits) | Pin Data (8 bits) | DMA0 DATA (32 bits) | Pin Data (8 bits) |
|                      |                     | U <sub>2</sub>    |                     | U <sub>2</sub>    |
|                      |                     | Y <sub>5</sub>    |                     | Y <sub>5</sub>    |
|                      |                     | V <sub>3</sub>    |                     | V <sub>3</sub>    |
|                      |                     | Y <sub>6</sub>    |                     | Y <sub>6</sub>    |
|                      |                     | U <sub>3</sub>    |                     | U <sub>3</sub>    |
|                      |                     | Y <sub>7</sub>    |                     | Y <sub>7</sub>    |

**Table 31-41:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=0 and SWAPEN=1

| DMACFG=1                                                    |                                                             |                   | DMACFG=0                                                    |                   |
|-------------------------------------------------------------|-------------------------------------------------------------|-------------------|-------------------------------------------------------------|-------------------|
| DMA0 DATA (32 bits)                                         | DMA1 DATA (32 bits)                                         | Pin Data (8 bits) | DMA0 DATA (32 bits)                                         | Pin Data (8 bits) |
| Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> | U <sub>1</sub>    | U <sub>1</sub> V <sub>1</sub> U <sub>0</sub> V <sub>0</sub> | U <sub>1</sub>    |
| Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> | Y <sub>3</sub>    | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | Y <sub>3</sub>    |
|                                                             |                                                             | V <sub>1</sub>    | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> | V <sub>1</sub>    |
|                                                             |                                                             | Y <sub>2</sub>    | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | Y <sub>2</sub>    |
|                                                             |                                                             | U <sub>0</sub>    |                                                             | U <sub>0</sub>    |
|                                                             |                                                             | Y <sub>1</sub>    |                                                             | Y <sub>1</sub>    |
|                                                             |                                                             | V <sub>0</sub>    |                                                             | V <sub>0</sub>    |
|                                                             |                                                             | Y <sub>0</sub>    |                                                             | Y <sub>0</sub>    |
|                                                             |                                                             | U <sub>3</sub>    |                                                             | U <sub>3</sub>    |
|                                                             |                                                             | Y <sub>7</sub>    |                                                             | Y <sub>7</sub>    |

**Table 31-41:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=0 and SWAPEN=1 (Continued)

| DMACFG=1            |                     |                   | DMACFG=0            |                   |
|---------------------|---------------------|-------------------|---------------------|-------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (8 bits) | DMA0 DATA (32 bits) | Pin Data (8 bits) |
|                     |                     | V <sub>3</sub>    |                     | V <sub>3</sub>    |
|                     |                     | Y <sub>6</sub>    |                     | Y <sub>6</sub>    |
|                     |                     | U <sub>2</sub>    |                     | U <sub>2</sub>    |
|                     |                     | Y <sub>5</sub>    |                     | Y <sub>5</sub>    |
|                     |                     | V <sub>2</sub>    |                     | V <sub>3</sub>    |
|                     |                     | Y <sub>4</sub>    |                     | Y <sub>4</sub>    |

**Table 31-42:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1, and SWAPEN=1

| DMACFG=1                                                    |                                                             |                   | DMACFG=0                                                    |                   |
|-------------------------------------------------------------|-------------------------------------------------------------|-------------------|-------------------------------------------------------------|-------------------|
| DMA0 DATA (32 bits)                                         | DMA1 DATA (32 bits)                                         | Pin Data (8 bits) | DMA0 DATA (32 bits)                                         | Pin Data (8 bits) |
| Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>3</sub>    | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>3</sub>    |
| Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | Y <sub>3</sub>    | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | Y <sub>3</sub>    |
|                                                             | V <sub>7</sub> V <sub>6</sub> V <sub>5</sub> V <sub>4</sub> | U <sub>3</sub>    | U <sub>3</sub> V <sub>3</sub> U <sub>2</sub> V <sub>2</sub> | U <sub>3</sub>    |
|                                                             | U <sub>7</sub> U <sub>6</sub> U <sub>5</sub> U <sub>4</sub> | Y <sub>2</sub>    | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | Y <sub>2</sub>    |
|                                                             |                                                             | V <sub>2</sub>    |                                                             | V <sub>2</sub>    |
|                                                             |                                                             | Y <sub>1</sub>    |                                                             | Y <sub>1</sub>    |
|                                                             |                                                             | U <sub>2</sub>    |                                                             | U <sub>2</sub>    |
|                                                             |                                                             | Y <sub>0</sub>    |                                                             | Y <sub>0</sub>    |
|                                                             |                                                             | V <sub>1</sub>    |                                                             | V <sub>1</sub>    |
|                                                             |                                                             | Y <sub>7</sub>    |                                                             | Y <sub>7</sub>    |
|                                                             |                                                             | U <sub>1</sub>    |                                                             | U <sub>1</sub>    |

**Table 31-42:** 8-bit Split Transmit Mode with SPLTEO=1, SUBSPLTODD=1, and SWAPEN=1 (Continued)

| DMACFG=1            |                     |                   | DMACFG=0            |                   |
|---------------------|---------------------|-------------------|---------------------|-------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (8 bits) | DMA0 DATA (32 bits) | Pin Data (8 bits) |
|                     |                     | Y <sub>6</sub>    |                     | Y <sub>6</sub>    |
|                     |                     | V <sub>0</sub>    |                     | V <sub>0</sub>    |
|                     |                     | Y <sub>5</sub>    |                     | Y <sub>5</sub>    |
|                     |                     | U <sub>0</sub>    |                     | U <sub>0</sub>    |
|                     |                     | Y <sub>4</sub>    |                     | Y <sub>4</sub>    |

### Configuring 10/12/14/16-Bit Transmit Mode with SPLTWRD = 0

For 16-bit split transmit mode, PACKEN is not valid. The EPPI always unpacks the 32-bit DMA data into two 16-bit words to transmit. For 10, 12, or 14-bit split transmit modes, the EPPI first unpacks the data in the same way as for 16-bit transmit mode, but transmits only the required number of LSBs.

**Table 31-43:** 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 0, and SWAPEN = 0

| DMACFG = 1                    |                               |                    | DMACFG = 0                    |                    |
|-------------------------------|-------------------------------|--------------------|-------------------------------|--------------------|
| DMA0 DATA (32 bits)           | DMA1 DATA (32 bits)           | Pin Data (16 bits) | DMA0 DATA (32 bits)           | Pin Data (16 bits) |
| Y <sub>1</sub> Y <sub>0</sub> | U <sub>0</sub> V <sub>0</sub> | V <sub>0</sub>     | U <sub>0</sub> V <sub>0</sub> | V <sub>0</sub>     |
| Y <sub>3</sub> Y <sub>2</sub> | U <sub>1</sub> V <sub>1</sub> | Y <sub>0</sub>     | Y <sub>1</sub> Y <sub>0</sub> | Y <sub>0</sub>     |
|                               |                               | U <sub>0</sub>     | U <sub>1</sub> V <sub>1</sub> | U <sub>0</sub>     |
|                               |                               | Y <sub>1</sub>     | Y <sub>3</sub> Y <sub>2</sub> | Y <sub>1</sub>     |
|                               |                               | V <sub>1</sub>     |                               | V <sub>1</sub>     |
|                               |                               | Y <sub>2</sub>     |                               | Y <sub>2</sub>     |
|                               |                               | U <sub>1</sub>     |                               | U <sub>1</sub>     |
|                               |                               | Y <sub>3</sub>     |                               | Y <sub>3</sub>     |

**Table 31-44:** 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 1, and SWAPEN = 0

| DMACFG = 1          |                     |                    | DMACFG = 0          |                    |
|---------------------|---------------------|--------------------|---------------------|--------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (16 bits) | DMA0 DATA (32 bits) | Pin Data (16 bits) |
| $Y_1Y_0$            | $V_1V_0$            | $V_0$              | $V_1V_0$            | $V_0$              |
| $Y_3Y_2$            | $U_1U_0$            | $Y_0$              | $Y_1Y_0$            | $Y_0$              |
|                     | $V_3V_2$            | $U_0$              | $U_1U_0$            | $U_0$              |
|                     | $U_3U_2$            | $Y_1$              | $Y_3Y_2$            | $Y_1$              |
|                     |                     | $V_1$              |                     | $V_1$              |
|                     |                     | $Y_2$              |                     | $Y_2$              |
|                     |                     | $U_1$              |                     | $U_1$              |
|                     |                     | $Y_3$              |                     | $Y_3$              |

**Table 31-45:** 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 0, and SWAPEN = 1

| DMACFG = 1          |                     |                    | DMACFG = 0          |                    |
|---------------------|---------------------|--------------------|---------------------|--------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (16 bits) | DMA0 DATA (32 bits) | Pin Data (16 bits) |
| $Y_1Y_0$            | $V_0U_0$            | $V_0$              | $V_0U_0$            | $V_0$              |
| $Y_3Y_2$            | $V_1U_1$            | $Y_1$              | $Y_1Y_0$            | $Y_1$              |
|                     |                     | $U_0$              | $V_1U_1$            | $U_0$              |
|                     |                     | $Y_0$              | $Y_3Y_2$            | $Y_0$              |
|                     |                     | $V_1$              |                     | $V_1$              |
|                     |                     | $Y_3$              |                     | $Y_3$              |
|                     |                     | $U_1$              |                     | $U_1$              |
|                     |                     | $Y_2$              |                     | $Y_2$              |

**Table 31-46:** 16-bit Split Transmit Mode with SPLTEO = 1, SUBSPLTODD = 1, and SWAPEN = 1

| DMACFG = 1          |                     |                    | DMACFG = 0          |                    |
|---------------------|---------------------|--------------------|---------------------|--------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (16 bits) | DMA0 DATA (32 bits) | Pin Data (16 bits) |
| $Y_1Y_0$            | $V_1V_0$            | $V_1$              | $V_1V_0$            | $V_1$              |
| $Y_3Y_2$            | $U_1U_0$            | $Y_1$              | $Y_1Y_0$            | $Y_1$              |
|                     | $V_3V_2$            | $U_1$              | $U_1U_0$            | $U_1$              |
|                     | $U_3U_2$            | $Y_0$              |                     | $Y_0$              |
|                     |                     | $V_0$              |                     | $V_0$              |
|                     |                     | $Y_3$              |                     | $Y_1$              |
|                     |                     | $U_0$              |                     | $U_0$              |
|                     |                     | $Y_2$              |                     | $Y_2$              |

### Configuring 16-Bit Split Transmit Mode with SPLTWRD = 1

For 16-bit split transmit mode, PACKEN is not valid. The EPPI always unpacks the 32-bit DMA data into two 16-bit words to transmit. The SPLTWRD bit is only valid when DLEN = 16 bits.

**Table 31-47:** 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 0, and SWAPEN = 0

| DMACFG = 1          |                     |                    | DMACFG = 0          |                    |
|---------------------|---------------------|--------------------|---------------------|--------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (16 bits) | DMA0 DATA (32 bits) | Pin Data (16 bits) |
| $Y_3Y_2Y_1Y_0$      | $U_1V_1U_0V_0$      | $V_0Y_0$           | $U_1V_1U_0V_0$      | $V_0Y_0$           |
| $Y_7Y_6Y_5Y_4$      | $U_3V_3U_2V_2$      | $U_0Y_1$           | $Y_3Y_2Y_1Y_0$      | $U_0Y_1$           |
|                     |                     | $V_1Y_2$           | $U_3V_3U_2V_2$      | $V_1Y_2$           |
|                     |                     | $U_1Y_3$           | $Y_7Y_6Y_5Y_4$      | $U_1Y_3$           |
|                     |                     | $V_2Y_4$           |                     | $V_2Y_4$           |
|                     |                     | $U_2Y_5$           |                     | $U_2Y_5$           |

**Table 31-47:** 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 0, and SWAPEN = 0 (Continued)

| DMACFG = 1          |                     |                               | DMACFG = 0          |                               |
|---------------------|---------------------|-------------------------------|---------------------|-------------------------------|
| DMA0 DATA (32 bits) | DMA1 DATA (32 bits) | Pin Data (16 bits)            | DMA0 DATA (32 bits) | Pin Data (16 bits)            |
|                     |                     | V <sub>3</sub> Y <sub>6</sub> |                     | V <sub>3</sub> Y <sub>6</sub> |
|                     |                     | U <sub>3</sub> Y <sub>7</sub> |                     | U <sub>3</sub> Y <sub>7</sub> |

**Table 31-48:** 16-bit Split Transmit Mode with SPLTWRD = 1, SUBSPLTODD = 1, and SWAPEN = 0

| DMACFG = 1                                                  |                                                             |                               | DMACFG = 0                                                  |                               |
|-------------------------------------------------------------|-------------------------------------------------------------|-------------------------------|-------------------------------------------------------------|-------------------------------|
| PRIMARY DMA DATA (32 bits)                                  | SECONDARY DMA DATA (32 bits)                                | Pin Data (16 bits)            | DMA0 DATA (32 bits)                                         | Pin Data (16 bits)            |
| Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>0</sub> Y <sub>0</sub> | V <sub>3</sub> V <sub>2</sub> V <sub>1</sub> V <sub>0</sub> | V <sub>0</sub> Y <sub>0</sub> |
| Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | U <sub>0</sub> Y <sub>1</sub> | Y <sub>3</sub> Y <sub>2</sub> Y <sub>1</sub> Y <sub>0</sub> | U <sub>0</sub> Y <sub>1</sub> |
|                                                             | V <sub>7</sub> V <sub>6</sub> V <sub>5</sub> V <sub>4</sub> | V <sub>1</sub> Y <sub>2</sub> | U <sub>3</sub> U <sub>2</sub> U <sub>1</sub> U <sub>0</sub> | V <sub>1</sub> Y <sub>2</sub> |
|                                                             | U <sub>7</sub> U <sub>6</sub> U <sub>5</sub> U <sub>4</sub> | U <sub>1</sub> Y <sub>3</sub> | Y <sub>7</sub> Y <sub>6</sub> Y <sub>5</sub> Y <sub>4</sub> | U <sub>1</sub> Y <sub>3</sub> |
|                                                             |                                                             | V <sub>2</sub> Y <sub>4</sub> |                                                             | V <sub>2</sub> Y <sub>4</sub> |
|                                                             |                                                             | U <sub>2</sub> Y <sub>5</sub> |                                                             | U <sub>2</sub> Y <sub>5</sub> |
|                                                             |                                                             | V <sub>3</sub> Y <sub>6</sub> |                                                             | V <sub>3</sub> Y <sub>6</sub> |
|                                                             |                                                             | U <sub>3</sub> Y <sub>7</sub> |                                                             | U <sub>3</sub> Y <sub>7</sub> |

## EPPI Programming Concepts

This section currently includes only SMPTE mode programming. The remaining content for this section is TBD.

### EPPI SMPTE Modes Programming

The programming model of SMPTE modes is similar to ITU Modes. All programming modes pertaining to ITU modes like XFRTYPE, FSCFG, FLDSEL and BLANKGEN hold true for SMPTE modes as well. The only difference is that since ITU modes use Y-Cr/Cb interleaved data and SMPTE use parallel Y-Cr/Cb



data, SPLTWRD should be set while operating in SMPTE modes. The following table describes the programming modes for different SMPTE formats.

**Table 31-49:** Programming Modes for SMPTE Formats

| SMPTE Format | SMPTE Channel Width | EPPI Input Bit Width                | EPPI Mode                    | Remarks                                               |
|--------------|---------------------|-------------------------------------|------------------------------|-------------------------------------------------------|
| 274M         | 8                   | 16<br>Cr/Cb - [15:8]<br>Y - [7:0]   | DLEN = 16bits<br>SPLTWRD = 1 | SIGNEXT not supported                                 |
|              | 10                  | 20<br>Cr/Cb - [19:10]<br>Y - [9:0]  | DLEN = 20bits<br>SPLTWRD = 1 | SIGNEXT extends each channel data to the 16b boundary |
|              | 12                  | 24<br>Cr/Cb - [23:12]<br>Y - [11:0] | DLEN = 24bits<br>SPLTWRD = 1 | SIGNEXT extends each channel data to the 16b boundary |
| 296M         | 8                   | 16<br>Cr/Cb - [15:8]<br>Y - [7:0]   | DLEN = 16bits<br>SPLTWRD = 1 | SIGNEXT not supported                                 |
|              | 10                  | 20<br>Cr/Cb - [19:10]<br>Y - [9:0]  | DLEN = 20bits<br>SPLTWRD = 1 | SIGNEXT extends each channel data to the 16b boundary |

Note: EPPI Clock at 74.25MHz for all the above modes

## EPPI Register Descriptions

EPPI (EPPI) contains the following registers.

**Table 31-50:** EPPI Module Registers

| Name      | Description                        |
|-----------|------------------------------------|
| EPPI_STAT | Status Register                    |
| EPPI_HCNT | Horizontal Transfer Count Register |
| EPPI_HDLY | Horizontal Delay Count Register    |
| EPPI_VCNT | Vertical Transfer Count Register   |

**Table 31-50:** EPPI Module Registers (Continued)

| Name           | Description                                                             |
|----------------|-------------------------------------------------------------------------|
| EPPI_VDLY      | Vertical Delay Count Register                                           |
| EPPI_FRAME     | Lines Per Frame Register                                                |
| EPPI_LINE      | Samples Per Line Register                                               |
| EPPI_CLKDIV    | Clock Divide Register                                                   |
| EPPI_CTL       | Control Register                                                        |
| EPPI_FS1_WLHB  | FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register |
| EPPI_FS1_PASPL | FS1 Period Register / EPPI Active Samples Per Line Register             |
| EPPI_FS2_WLVB  | FS2 Width Register / EPPI Lines Of Vertical Blanking Register           |
| EPPI_FS2_PALPF | FS2 Period Register / EPPI Active Lines Per Field Register              |
| EPPI_IMSK      | Interrupt Mask Register                                                 |
| EPPI_ODDCLIP   | Clipping Register for ODD (Chroma) Data                                 |
| EPPI_EVENCLIP  | Clipping Register for EVEN (Luma) Data                                  |
| EPPI_FS1_DLY   | Frame Sync 1 Delay Value                                                |
| EPPI_FS2_DLY   | Frame Sync 2 Delay Value                                                |
| EPPI_CTL2      | Control Register 2                                                      |

## Status Register

The EPPI\_STAT register contains bits that provide information about the current operating state of the EPPI.

**EPPI\_STAT: Status Register - R/W**

Reset = 0x0000 0000

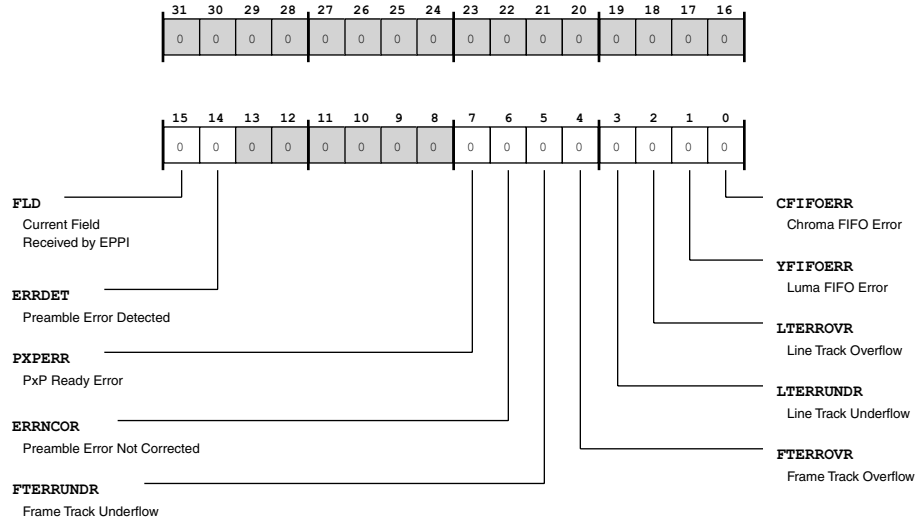


**Figure 31-15:** EPPI\_STAT Register Diagram

**Table 31-51:** EPPI\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15      | FLD      | Current Field Received by EPPI.<br>The <code>EPPI_STAT.FLD</code> bit indicates whether the current field being received by the PPI is Field 1 (if clear) or Field 2 (if set).                                                                                          |
|         |          | 0      Field 1                                                                                                                                                                                                                                                          |
|         |          | 1      Field 2                                                                                                                                                                                                                                                          |
| 14      | ERRDET   | Preamble Error Detected.<br>The <code>EPPI_STAT.ERRDET</code> bit is useful only in ITU receive modes and indicates if an error has been detected in the status word of EAV or SAV sequences (if set) or not (if clear).                                                |
|         |          | 0      No preamble error detected                                                                                                                                                                                                                                       |
|         |          | 1      Preamble error detected                                                                                                                                                                                                                                          |
| 7       | PXPERR   | PxP Ready Error.<br>The <code>EPPI_STAT.PXPERR</code> bit is valid only in the RX mode. This bit indicates whether the incoming PPI data overflows the PxP interface (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it. |

**Table 31-51:** EPPI\_STAT Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                                                                    |
|---------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | ERRNCOR   | Preamble Error Not Corrected.<br>The <code>EPPI_STAT.ERRNCOR</code> bit is useful only in the ITU receive modes and indicates if an error in the status word of EAV or SAV sequences can not be cleared (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it. |
|         |           | 0      No uncorrected preamble error has occurred                                                                                                                                                                                                                                                          |
|         |           | 1      Preamble error detected but not corrected                                                                                                                                                                                                                                                           |
| 5       | FTERRUNDR | Frame Track Underflow.<br>The <code>EPPI_STAT.FTERRUNDR</code> bit indicates whether a frame track underflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.                                                                         |
|         |           | 0      No Error Detected                                                                                                                                                                                                                                                                                   |
|         |           | 1      Error Occurred                                                                                                                                                                                                                                                                                      |
| 4       | FTERROVR  | Frame Track Overflow.<br>The <code>EPPI_STAT.FTERROVR</code> bit indicates whether a frame track overflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.                                                                            |
|         |           | 0      No Error Detected                                                                                                                                                                                                                                                                                   |
|         |           | 1      Error Occurred                                                                                                                                                                                                                                                                                      |
| 3       | LTERRUNDR | Line Track Underflow.<br>The <code>EPPI_STAT.LTERRUNDR</code> bit indicates whether a line track underflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.                                                                           |
|         |           | 0      No Error Detected                                                                                                                                                                                                                                                                                   |
|         |           | 1      Error Occurred                                                                                                                                                                                                                                                                                      |

**Table 31-51:** EPPI\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                 |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | LTERROVR | Line Track Overflow.<br>The EPPI_STAT.LTERROVR bit indicates whether a line track overflow error has occurred (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.                                                                                                        |
|         |          | 0 No Error Detected                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                                                        |
| 1       | YFIFOERR | Luma FIFO Error.<br>For RX modes, the EPPI_STAT.YFIFOERR bit indicates whether the Luma FIFO has overflowed (if set) or not (if clear). For TX modes, this bit indicates whether the Luma FIFO has underflowed (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it.       |
|         |          | 0 No Error Detected                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                                                        |
| 0       | CFIFOERR | Chroma FIFO Error.<br>For RX modes, the EPPI_STAT.CFIFOERR bit indicates whether the Chroma FIFO has overflowed (if set) or not (if clear). For TX modes, this bit indicates whether the Chroma FIFO has underflowed (if set) or not (if clear). This bit is sticky and must be cleared by software by writing 1 to it. |
|         |          | 0 No Error Detected                                                                                                                                                                                                                                                                                                     |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                                                        |

## Horizontal Transfer Count Register

The EPPI\_HCNT register holds the number of samples to read in or write out per line, after EPPI\_HDLY number of cycles have expired since the assertion of EPPI\_FS1. Any write to the EPPI\_LINE register modifies the EPPI\_HCNT register, but any write to EPPI\_HCNT does not affect the EPPI\_LINE register value. So, the EPPI\_HCNT register should be programmed after the EPPI\_LINE register.

EPPI\_HCNT: Horizontal Transfer Count Register - R/W

Reset = 0x0000 0000

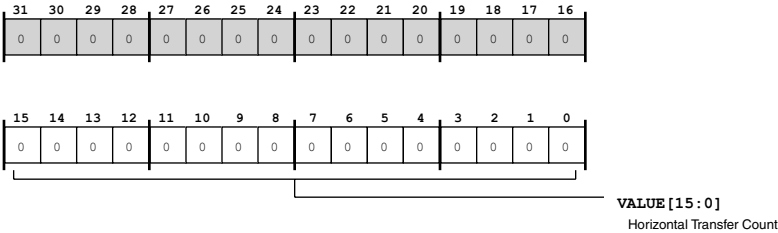


Figure 31-16: EPPI\_HCNT Register Diagram

Table 31-52: EPPI\_HCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Horizontal Transfer Count.<br>The EPPI_HCNT.VALUE holds the number of samples to read in or write out per line, after EPPI_HDLY number of cycles have expired since the last assertion of EPPI_FS1. |

Horizontal Delay Count Register

The EPPI\_HDLY register contains the number of clock cycles to delay after the assertion of EPPI\_FS1 is detected before starting to read or write data.

EPPI\_HDLY: Horizontal Delay Count Register - R/W

Reset = 0x0000 0000

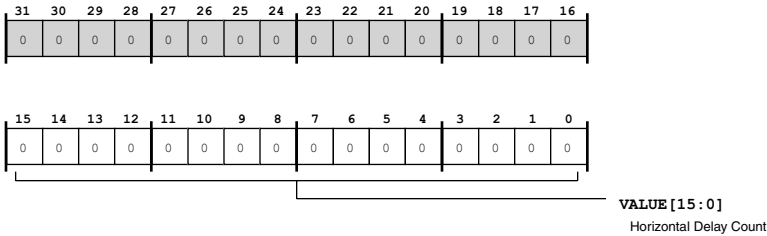


Figure 31-17: EPPI\_HDLY Register Diagram

**Table 31-53:** EPPI\_HDLY Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                        |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Horizontal Delay Count.<br>The EPPI_HDLY.VALUE holds the number of EPPI_CLK cycles to delay after assertion of EPPI_FS1 before starting to read or write data. |

## Vertical Transfer Count Register

The EPPI\_VCNT register holds the number of lines to read in or write out, after EPPI\_VDLY number of lines from the start of frame. Any write to the EPPI\_FRAME register modifies the EPPI\_VCNT register, but any write to EPPI\_VCNT does not affect the EPPI\_FRAME register value. So, the EPPI\_VCNT register should be programmed after the EPPI\_FRAME register.

**EPPI\_VCNT: Vertical Transfer Count Register - R/W**

Reset = 0x0000 0000

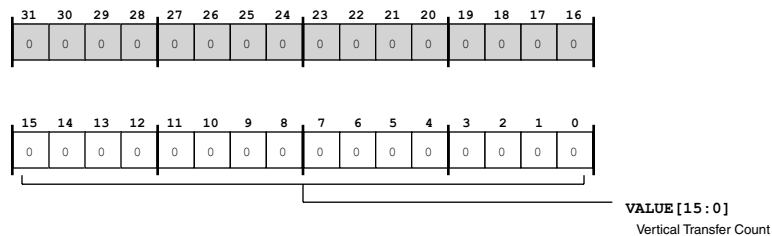


**Figure 31-18:** EPPI\_VCNT Register Diagram

**Table 31-54:** EPPI\_VCNT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Vertical Transfer Count.<br>The EPPI_VCNT.VALUE holds the number of lines to read in or write out, after EPPI_VDLY number of lines from the start of frame. |

## Vertical Delay Count Register

The EPPI\_VDLY register contains the number of lines to wait after the start of a new frame before starting to read/transmit data.

EPPI\_VDLY: Vertical Delay Count Register - R/W

Reset = 0x0000 0000

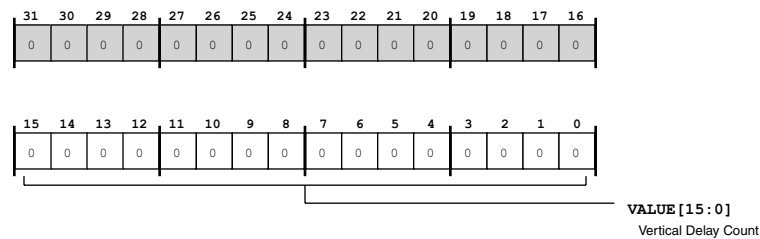


Figure 31-19: EPPI\_VDLY Register Diagram

Table 31-55: EPPI\_VDLY Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                              |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Vertical Delay Count.<br>The EPPI_VDLY.VALUE holds the number of lines to wait after the start of a new frame before starting to read/transmit data. |

Lines Per Frame Register

The EPPI\_FRAME register tracks the frame track overflow and underflow errors. This register should be programmed with the number of lines expected per frame. Any write to the EPPI\_FRAME register will also write the same value to the EPPI\_VCNT register, but any write to EPPI\_VCNT does not affect the EPPI\_FRAME register value. So, the EPPI\_FRAME register should be programmed before the EPPI\_VCNT register.

EPPI\_FRAME: Lines Per Frame Register - R/W

Reset = 0x0000 0000

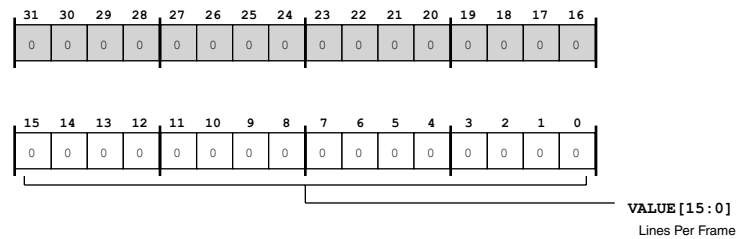


Figure 31-20: EPPI\_FRAME Register Diagram



**Table 31-56:** EPPI\_FRAME Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                        |
|---------|----------|------------------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Lines Per Frame.<br>The EPPI_FRAME.VALUE holds the number of lines expected per frame of data. |

## Samples Per Line Register

The EPPI\_LINE register tracks the line track overflow and underflow Errors. This register should be programmed with the number of samples expected per line. Any write to the EPPI\_LINE register will also write the same value to the EPPI\_HCNT register, but any write to EPPI\_HCNT does not affect the EPPI\_LINE register value. So, the EPPI\_LINE register should be programmed before the EPPI\_HCNT register.

### EPPI\_LINE: Samples Per Line Register - R/W

Reset = 0x0000 0000

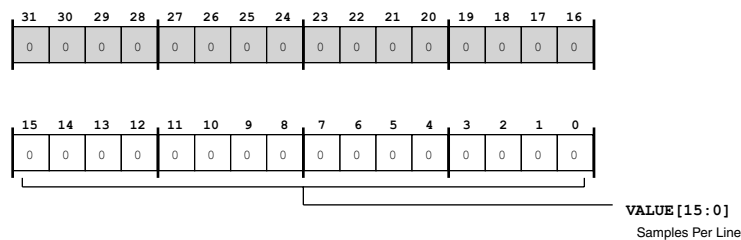


**Figure 31-21:** EPPI\_LINE Register Diagram

**Table 31-57:** EPPI\_LINE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------|
| 15:0    | VALUE    | Samples Per Line.<br>The EPPI_LINE.VALUE holds the number of samples expected per line. |

## Clock Divide Register

The EPPI\_CLKDIV register provides the divisor for EPPI internal clock generation. The generated clock frequency is given by following formula:

$$\text{EPPI\_CLK} = (\text{SCLK}) / (2 * (\text{EPPI\_CLKDIV} + 1))$$

Note that a value of 0xFFFF is invalid for the EPPI\_CLKDIV register.

EPPI\_CLKDIV: Clock Divide Register - R/W

Reset = 0x0000 0000

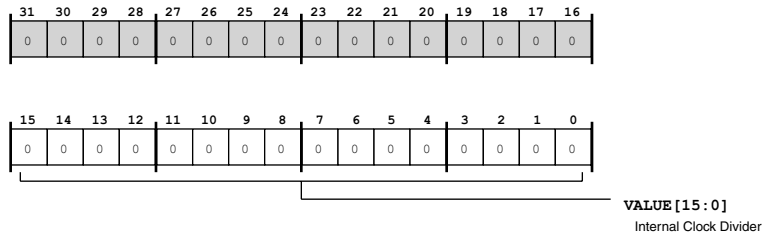


Figure 31-22: EPPI\_CLKDIV Register Diagram

Table 31-58: EPPI\_CLKDIV Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 15:0    | VALUE    | Internal Clock Divider. |

Control Register

The EPPI\_CTL register configures the EPPI for operating mode, control signal polarities, and data width of the port.

### EPPI\_CTL: Control Register - R/W

Reset = 0x0000 0000

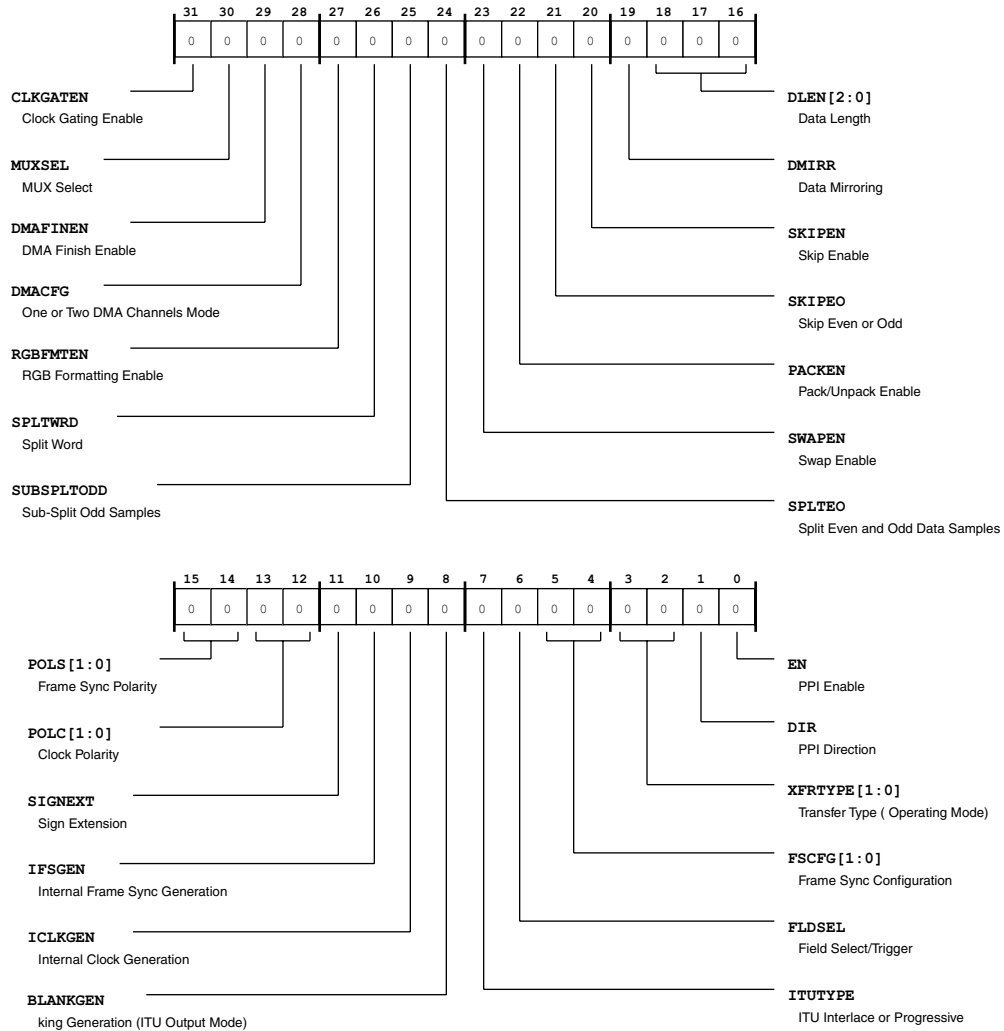


Figure 31-23: EPPI\_CTL Register Diagram

**Table 31-59:** EPPI\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                   |                            |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------|
| 31      | CLKGATEN | Clock Gating Enable.<br>The EPPI_CTL.CLKGATEN bit enables using the EPPI_FS3 pin as a clock gating pin. When EPPI_CTL.CLKGATEN is set, the EPPI_FS3 pin acts as a clock gating signal, and both the internal and external clock are gated. Note that the EPPI_FS3 pin gating signal is active low, and the EPPI_CTL.CLKGATEN selection is ignored if EPPI_CTL.MUXSEL is set or EPPI_CTL.FSCFG equals 0x3. |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                         | Disable                    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                         | Enable                     |
| 30      | MUXSEL   | MUX Select.<br>The EPPI_CTL.MUXSEL bit enables multiplexing of a primary and alternate camera using the EPPI main data and clock lines. For more information on this feature, see the EPPI functional description.                                                                                                                                                                                        |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                         | Normal Operation           |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                         | Multiplexed Operation      |
| 29      | DMAFINEN | DMA Finish Enable.<br>The EPPI_CTL.DMAFINEN bit selects whether or not the EPPI sends a finish command (010) through the DDE COMMAND line soon after a frame/line is received completely.                                                                                                                                                                                                                 |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                         | No Finish Command          |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                         | Enable Send Finish Command |
| 28      | DMACFG   | One or Two DMA Channels Mode.<br>The EPPI_CTL.DMACFG bit is valid only if EPPI_CTL.SPLTE0 is set. If EPPI_CTL.DMACFG is set, the EPPI uses two DMA channels. And, if EPPI_CTL.DMACFG is cleared, the EPPI uses only one DMA channel.                                                                                                                                                                      |                            |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                         | PPI uses one DMA Channel   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                         | PPI uses two DMA Channels  |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                             |
|---------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| 27      | RGBFMTEN   | RGB Formatting Enable.<br>For 16- or 18-bit transmit modes only, the EPPI_CTL.RGBFMTEN bit enables conversion of RGB888 from memory into RGB666 output data (18-bit transmit) or enables conversion of RGB888 from memory into RGB565 output data (16-bit transmit). Note that EPPI_CTL.SPLTE0 and EPPI_CTL.RGBFMTEN should never be set simultaneously.                                                                                                                                                                                                                                                    |                                             |
|         |            | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Disable RGB Formatted Output                |
|         |            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Enable RGB Formatted Output                 |
| 26      | SPLTWRD    | Split Word.<br>The EPPI_CTL.SPLTWRD bit selects split word data placement when the data length (EPPI_CTL.DLEN) selects 16-, 20-, or 24-bit data words. For all other EPPI_CTL.SPLTWRD values, the set or clear selections for EPPI_CTL.SPLTWRD produce the same result (act as though EPPI_CTL.SPLTWRD is cleared). For EPPI_CTL.SPLTWRD set, the EPPI_CTL.DLEN values below result in the following combinations of split words:<br><div>DLEN      Cr/Cb data      Y data<br/>16   PPI_DATA[15:8]   PPI_DATA[7:0]<br/>20   PPI_DATA[19:10]   PPI_DATA[9:0]<br/>24   PPI_DATA[23:12]   PPI_DATA[11:0]</div> |                                             |
|         |            | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | PPI_DATA has (DLEN-1) bits of Y or Cr or Cb |
|         |            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | PPI_DATA contains 2 elements per word       |
| 25      | SUBSPLTODD | Sub-Split Odd Samples.<br>The EPPI_CTL.SUBSPLTODD bit is valid only if EPPI_CTL.SPLTE0 is set. If EPPI_CTL.SUBSPLTODD is set, the EPPI sub-splits the odd sub-stream, and packs them separately.                                                                                                                                                                                                                                                                                                                                                                                                            |                                             |
|         |            | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Disable                                     |
|         |            | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | Enable                                      |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 24      | SPLTEO   | Split Even and Odd Data Samples.<br>If EPPI_CTL.SPLTEO is set, the EPPI splits the incoming data stream into two sub-streams, an even stream and an odd stream, and packs them separately.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0 Do Not Split Samples                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Split Even/Odd Samples                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 23      | SWAPEN   | Swap Enable.<br>The EPPI_CTL.SWAPEN selects whether or not to swap the order of the first data (most-significant bits versus least-significant bits) of the DMA word.<br>For receive modes, the EPPI puts the first data in the most significant bits (if set) or puts the first data in the least significant bits (if cleared) of the DMA word.<br>For transmit modes, the EPPI transmits the most significant bits in the DMA word as the first data (if set) or transmits the least significant bits in the DMA word as the first data (if cleared).                                                                                                               |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 22      | PACKEN   | Pack/Unpack Enable.<br>The EPPI_CTL.PACKEN select whether or not packing is enabled (for receive modes) and unpacking is enabled (for transmit modes).<br>When this bit is set, EPPI transfer DMA is 32-bits wide. When this bit is cleared and the EPPI_CTL.DLEN is less than or equal to 16 bits, EPPI transfer DMA is 16-bits wide.<br>For receive modes, if this bit is set, then the EPPI packs the incoming data into 32-bit words. If this bit is cleared, then the EPPI does not do any packing.<br>For transmit modes, if this bit is set, then the EPPI always unpacks the 32-bit data from DMA. If this bit is not set, the EPPI does not do any unpacking. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name          | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|---------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|----------------------|----------------------|----------------------|-------|--|--|----|----|---|----|----|---|------|------|------|---|---|----|---|---|----|
| 21      | SKIPEO            | Skip Even or Odd.<br>The EPPI_CTL.SKIPEO bit selects whether even (if set) or odd (if cleared) samples are skipped if sample skipping is enabled (EPPI_CTL.SKIPEN is set). This feature only is useful for receive modes.                                                                                                                                                                                                                                                                        |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|         |                   | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Skip Odd Samples       |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|         |                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Skip Even Samples      |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 20      | SKIPEN            | Skip Enable.<br>The EPPI_CTL.SKIPEN bit enables skipping alternate samples. This feature only is useful for receive modes.                                                                                                                                                                                                                                                                                                                                                                       |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|         |                   | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | No Samples Skipping    |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|         |                   | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Skip Alternate Samples |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 19      | DMIRR             | Data Mirroring.<br>The EPPI_CTL.DMIRR field enables mirroring (bit reversing) the data coming in or going out on the EPPI data pins.<br><table><tr><td>Pin Data</td><td>PPI Data (DAT_MRR=0)</td><td>PPI Data (DAT_MRR=1)</td></tr><tr><td colspan="3">-----</td></tr><tr><td>15</td><td>15</td><td>0</td></tr><tr><td>14</td><td>14</td><td>1</td></tr><tr><td>....</td><td>....</td><td>....</td></tr><tr><td>1</td><td>1</td><td>14</td></tr><tr><td>0</td><td>0</td><td>15</td></tr></table> |                        | Pin Data             | PPI Data (DAT_MRR=0) | PPI Data (DAT_MRR=1) | ----- |  |  | 15 | 15 | 0 | 14 | 14 | 1 | .... | .... | .... | 1 | 1 | 14 | 0 | 0 | 15 |
|         |                   | Pin Data                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | PPI Data (DAT_MRR=0)   | PPI Data (DAT_MRR=1) |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
|         |                   | -----                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 15      | 15                | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 14      | 14                | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| ....    | ....              | ....                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 1       | 1                 | 14                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 0       | 0                 | 15                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 0       | No Data Mirroring |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |
| 1       | Data Mirroring    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                        |                      |                      |                      |       |  |  |    |    |   |    |    |   |      |      |      |   |   |    |   |   |    |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                  |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18:16   | DLEN     | <p>Data Length.</p> <p>The EPPI_CTL.DLEN bits select the data length for the EPPI. Note that the 20 bits data length selection is valid only for SMPTE modes (EPPI_CTL.SPLTWRD set).</p> |
|         |          | 0 8 bits                                                                                                                                                                                 |
|         |          | 1 10 bits                                                                                                                                                                                |
|         |          | 2 12 bits                                                                                                                                                                                |
|         |          | 3 14 bits                                                                                                                                                                                |
|         |          | 4 16 bits                                                                                                                                                                                |
|         |          | 5 18 bits                                                                                                                                                                                |
|         |          | 6 20 bits                                                                                                                                                                                |
|         |          | 7 24 bits                                                                                                                                                                                |
| 15:14   | POLS     | <p>Frame Sync Polarity.</p> <p>The EPPI_CTL.POLS selects whether the frame syncs' polarity is active low versus active high.</p>                                                         |
|         |          | 0 FS1 and FS2 are active high                                                                                                                                                            |
|         |          | 1 FS1 is active low. FS2 is active high                                                                                                                                                  |
|         |          | 2 FS1 is active high. FS2 is active low                                                                                                                                                  |
|         |          | 3 FS1 and FS2 are active low                                                                                                                                                             |



**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                         |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 13:12   | POLC     | Clock Polarity.<br>The EPPI_CTL.POLC selects the rising versus falling edge for sampling data and sampling/driving syncs.                                                                                                       |
|         |          | 0<br>Clock/Sync polarity mode 0<br>For Receive mode: Sample data on falling edge and sample/drive syncs on falling edge<br>For Transmit mode: Drive data on rising edge and sample/drive syncs on rising edge                   |
|         |          | 1<br>Clock/Sync polarity mode 1<br>For Receive mode: Sample data on falling edge and sample/drive syncs on rising edge<br>For Transmit mode: Drive data on rising edge and sample/drive syncs on falling edge                   |
|         |          | 2<br>Clock/Sync polarity mode 2<br>For Receive mode: Sample data on rising edge and sample/drive syncs on falling edge<br>For Transmit mode: Drive data on falling edge and sample/drive syncs on rising edge                   |
|         |          | 3<br>Clock/Sync polarity mode 3<br>For Receive mode: Sample data on rising edge and sample/drive syncs on rising edge<br>For Transmit mode: Drive data on falling edge and sample/drive syncs on falling edge                   |
| 11      | SIGNEXT  | Sign Extension.<br>The EPPI_CTL.SIGNEXT select whether (for receive modes when EPPI_CTL.DLEN selecting 16 bit data length) the data is sign extended or zero filled. Not that EPPI_CTL.SPLTWRD is removed from this shared bit. |
|         |          | 0<br>Zero Filled                                                                                                                                                                                                                |
|         |          | 1<br>Sign Extended                                                                                                                                                                                                              |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                      |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10      | IFSGEN   | Internal Frame Sync Generation.<br>The EPPI_CTL.IFSGEN bit selects whether the frame syncs are generated internally or are supplied by an external device.                                                                                                                                                                                                                                   |
|         |          | 0 External Frame Sync                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 Internal Frame Sync                                                                                                                                                                                                                                                                                                                                                                        |
| 9       | ICLKGEN  | Internal Clock Generation.<br>The EPPI_CTL.ICLKGEN bit selects whether the EPPI_CLK is generated internally or is supplied by an external device.                                                                                                                                                                                                                                            |
|         |          | 0 External Clock                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1 Internal Clock                                                                                                                                                                                                                                                                                                                                                                             |
| 8       | BLANKGEN | Blanking Generation (ITU Output Mode).<br>The EPPI_CTL.BLANKGEN enables ITU output with internal blanking. In GP 8, 10 transmit bit modes (when EPPI_CTL.SPLTWRD is cleared) and 16-, 20-, and 24-bit transmit modes (when EPPI_CTL.SPLTWRD is set), EPPI_CTL.BLANKGEN selects whether or not the EPPI generates blanking and generates preamble and insertion with active data from memory. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                     |
| 7       | ITUTYPE  | ITU Interlace or Progressive.<br>The EPPI_CTL.ITUTYPE selects interlaced or progressive operation for ITU656 mode. This selection is valid for both TX and RX modes.                                                                                                                                                                                                                         |
|         |          | 0 Interlaced                                                                                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Progressive                                                                                                                                                                                                                                                                                                                                                                                |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                       |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | FLDSEL   | Field Select/Trigger.<br>The EPPI_CTL.FLDSEL bits configure the EPPI field and trigger selection. These are valid for GP modes (EPPI_CTL.XFRTYPE = 0x3) and ITU656 active video mode (EPPI_CTL.XFRTYPE cleared).                                              |
|         |          | 0<br>Field Mode 0<br>Read Field 1 (for ITU656 active video mode). Set internal trigger (for GP RX mode). FS3 is toggled on FS2 assertion followed by FS1 assertion (when EPPI_CTL.FSCFG selects sync mode 3 and EPPI_CTL.IFSGEN selects internal frame sync). |
|         |          | 1<br>Field Mode 1<br>Read Field 1 and Field 2 (ITU656 active video mode). Set external trigger (GP RX mode). FS3 is toggled on FS2 assertion (when EPPI_CTL.FSCFG selects sync mode 3 and EPPI_CTL.IFSGEN selects internal frame sync).                       |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                             |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| 5:4     | FSCFG    | Frame Sync Configuration.<br>The EPPI_CTL.FSCFG bits configure the EPPI frame syncs. These are valid only for GP modes (EPPI_CTL.XFRTYPE =0x3). The output of the frames syncs also depends on whether the EPPI transfer direction is transmit and the EPPI is in ITU output mode (EPPI_CTL.BLANKGEN is set).                                                                                                                  |                                                                                                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                              | Sync Mode 0<br>FS0 driven in GP mode. FS0 not driven in ITU output mode.                                                    |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                              | Sync Mode 1<br>FS1 driven in GP mode. HSYNC driven on FS1 in ITU output mode.                                               |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                              | Sync Mode 2<br>FS2 driven in GP mode. HSYNC driven on FS1 and VSYNC driven on FS1 in ITU output mode.                       |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                              | Sync Mode 3<br>FS3 driven in GP mode. HSYNC driven on FS1, VSYNC driven on FS2, and FIELD driven on FS3 in ITU output mode. |
| 3:2     | XFRTYPE  | Transfer Type ( Operating Mode).<br>The EPPI_CTL.XFRTYPE bits select the EPPI operating mode. In receive mode (EPPI_CTL.DIR cleared), the EPPI modes include ITU656 active video only mode, ITU656 entire field mode, ITU656 vertical blanking only mode, and non-ITU656 mode (GP mode). For transmit mode (EPPI_CTL.DIR set), the EPPI_CTL.XFRTYPE bits have no effect, and the EPPI (in transmit mode) is always in GP mode. |                                                                                                                             |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                              | ITU656 Active Video Only Mode                                                                                               |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                              | ITU656 Entire Field Mode                                                                                                    |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                              | ITU656 Vertical Blanking Only Mode                                                                                          |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                              | Non-ITU656 Mode (GP Mode)                                                                                                   |

**Table 31-59:** EPPI\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|
| 1       | DIR      | PPI Direction.<br>The EPPI_CTL.DIR bit selects whether the EPPI is in receive mode (if cleared) or in transmit mode (if set). |
|         |          | 0      Receive Mode                                                                                                           |
|         |          | 1      Transmit Mode                                                                                                          |
| 0       | EN       | PPI Enable.<br>The EPPI_CTL.EN bit enables or disables the EPPI.                                                              |
|         |          | 0      Disable                                                                                                                |
|         |          | 1      Enable                                                                                                                 |

## FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register

The EPPI\_FS1\_WLHB register's content varies depending on whether the EPPI is in GP1/2/3 FS modes or in GP transmit mode.

In GP 1, 2 or 3 FS modes, EPPI\_FS1\_WLHB is used for the generation of Frame Sync 1. The register contains the width required for EPPI\_FS1 based on the EPPI\_CLK clock.

In GP transmit mode with EPPI\_CTL.BLANKGEN set, this register contains the number of samples of horizontal blanking per line. When used for blanking generation, only the lower 16 bits are valid.

Note that a value of 0 for the EPPI\_FS1\_WLHB register is illegal. If programmed as 0, the EPPI regards the EPPI\_FS1\_WLHB as containing 1.

### EPPI\_FS1\_WLHB: FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register - R/W

Reset = 0x0000 0000

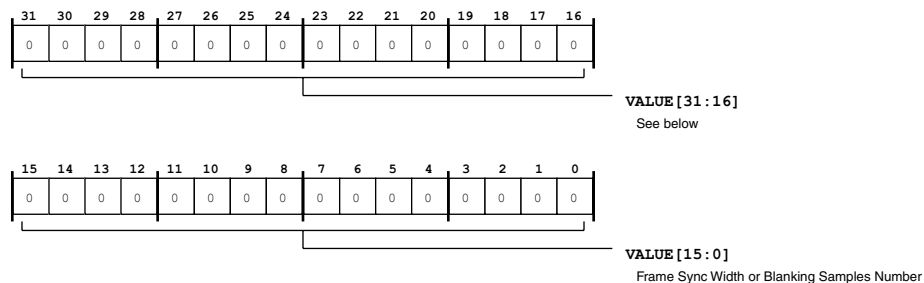


**Figure 31-24:** EPPI\_FS1\_WLHB Register Diagram

Table 31-60: EPPI\_FS1\_WLHB Register Fields

| Bit No. | Bit Name | Description/Enumeration                      |
|---------|----------|----------------------------------------------|
| 31:0    | VALUE    | Frame Sync Width or Blanking Samples Number. |

FS1 Period Register / EPPI Active Samples Per Line Register

The EPPI\_FS1\_PASPL register content varies depending on whether the EPPI is in GP1/2/3 FS modes or in GP transmit mode.

In GP 1, 2, or 3 FS modes, EPPI\_FS1\_PASPL is used for the generation of Frame Sync 1. The register contains the period required for EPPI\_FS1 based on the EPPI\_CLK clock.

In GP transmit mode with EPPI\_CTL.BLANKGEN set, this register contains the number of samples of active video or vertical blanking samples per line. When used for blanking generation, only the lower 16 bits are valid.

Note that a value of 0 for this register is illegal. If programmed as 0, the EPPI regards the EPPI\_FS1\_PASPL as containing 1.

EPPI\_FS1\_PASPL: FS1 Period Register / EPPI Active Samples Per Line Register - R/W

Reset = 0x0000 0000

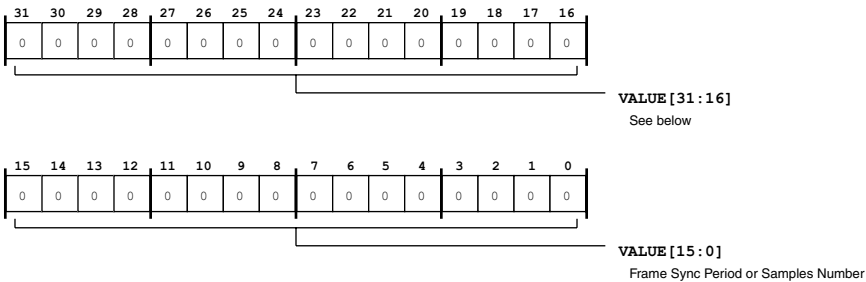


Figure 31-25: EPPI\_FS1\_PASPL Register Diagram

Table 31-61: EPPI\_FS1\_PASPL Register Fields

| Bit No. | Bit Name | Description/Enumeration              |
|---------|----------|--------------------------------------|
| 31:0    | VALUE    | Frame Sync Period or Samples Number. |

## FS2 Width Register / EPPI Lines Of Vertical Blanking Register

The EPPI\_FS2\_WLVB register content varies depending on whether the EPPI is in GP2/3 FS modes or in GP transmit mode.

In GP 2 or 3 FS modes, EPPI\_FS2\_WLVB is used for the generation of Frame Sync 2. The register contains the width required for EPPI\_FS2 based on the EPPI\_CLK clock.

In GP transmit mode with EPPI\_CTL.BLANKGEN set, this register contains the number or lines of vertical blanking.

Note that for progressive video, EPPI\_FS2\_WLVB.F2VBBD and EPPI\_FS2\_WLVB.F2VBAD are ignored.

**EPPI\_FS2\_WLVB: FS2 Width Register / EPPI Lines Of Vertical Blanking Register - R/W**

Reset = 0x0000 0000

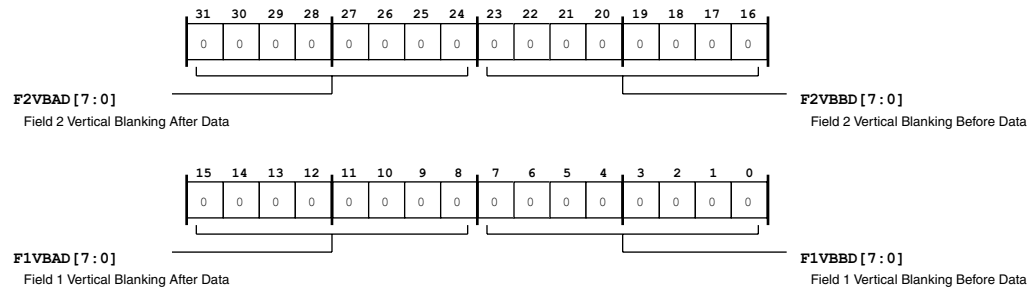


**Figure 31-26:** EPPI\_FS2\_WLVB Register Diagram

**Table 31-62:** EPPI\_FS2\_WLVB Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                            |
|---------|----------|----------------------------------------------------------------------------------------------------|
| 31:24   | F2VBAD   | Field 2 Vertical Blanking After Data.<br>The number of lines of vertical blanking after field 2.   |
| 23:16   | F2VBBD   | Field 2 Vertical Blanking Before Data.<br>The number of lines of vertical blanking before field 2. |
| 15:8    | F1VBAD   | Field 1 Vertical Blanking After Data.<br>The number of lines of vertical blanking after field 1.   |
| 7:0     | F1VBBD   | Field 1 Vertical Blanking Before Data.<br>The number of lines of Vertical blanking before field 1. |

## FS2 Period Register / EPPI Active Lines Per Field Register

The EPPI\_FS2\_PALPF register content varies depending on whether the EPPI is in GP2/3 FS modes or in GP transmit mode.

In GP 2 or 3 FS modes, EPPI\_FS2\_PALPF is used for the generation of Frame Sync 2. This register contains the period required for EPPI\_FS2 based on the EPPI\_CLK clock.

In GP transmit mode with EPPI\_CTL.BLANKGEN set, this register contains the number of lines of active video per field.

Note that a value of 0 for EPPI\_FS2\_PALPF.F1ACT or EPPI\_FS2\_PALPF.F2ACT is illegal. If either is programmed as 0, the EPPI regard the 0 value fields as containing 1.

Also note that for progressive video, EPPI\_FS2\_PALPF.F2ACT is ignored.

EPPI\_FS2\_PALPF: FS2 Period Register / EPPI Active Lines Per Field Register - R/W

Reset = 0x0000 0000

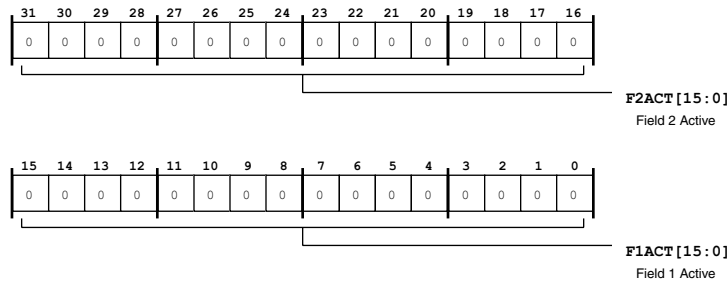


Figure 31-27: EPPI\_FS2\_PALPF Register Diagram

Table 31-63: EPPI\_FS2\_PALPF Register Fields

| Bit No. | Bit Name | Description/Enumeration                                           |
|---------|----------|-------------------------------------------------------------------|
| 31:16   | F2ACT    | Field 2 Active.<br>The number of lines of Active Data in Field 2. |
| 15:0    | F1ACT    | Field 1 Active.<br>The number of lines of Active Data in Field 1. |

## Interrupt Mask Register

The EPPI\_IMSK permits masking (if associated bit is set) of EPPI error interrupts for YFIFO underflow or overflow, CFIFO underflow or overflow, line track overflow error, line track underflow error, frame track overflow error, frame track underflow error, and ERR\_NCOR (ITU preamble error not corrected). These conditions are flagged in the EPPI\_STAT register and cleared by write-1-to-clear.



**EPPI\_IMSK: Interrupt Mask Register - R/W**

Reset = 0x0000 0000

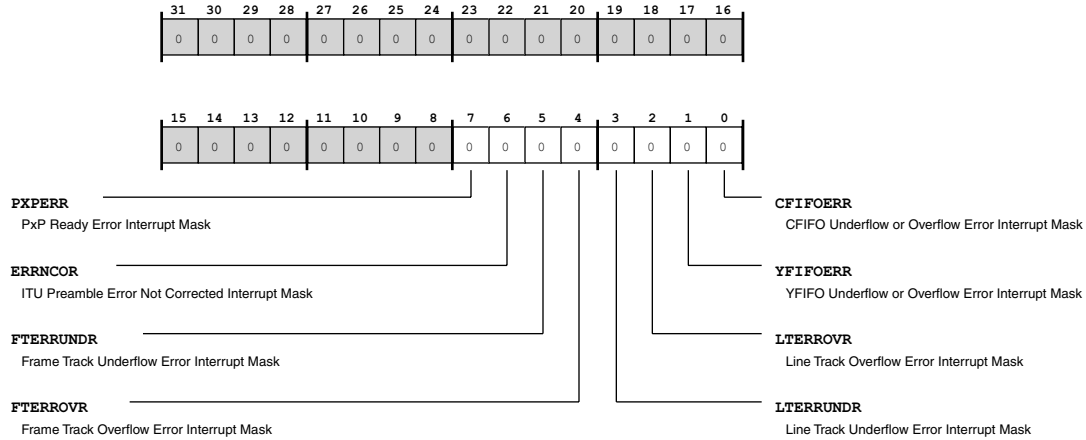


**Figure 31-28:** EPPI\_IMSK Register Diagram

**Table 31-64:** EPPI\_IMSK Register Fields

| Bit No. | Bit Name  | Description/Enumeration                          |
|---------|-----------|--------------------------------------------------|
| 7       | PXPERR    | PxP Ready Error Interrupt Mask.                  |
|         |           | 0 Unmask Interrupt                               |
|         |           | 1 Mask Interrupt                                 |
| 6       | ERRNCOR   | ITU Preamble Error Not Corrected Interrupt Mask. |
|         |           | 0 Unmask Interrupt                               |
|         |           | 1 Mask Interrupt                                 |
| 5       | FTERRUNDR | Frame Track Underflow Error Interrupt Mask.      |
|         |           | 0 Unmask Interrupt                               |
|         |           | 1 Mask Interrupt                                 |
| 4       | FTERROVR  | Frame Track Overflow Error Interrupt Mask.       |
|         |           | 0 Unmask Interrupt                               |
|         |           | 1 Mask Interrupt                                 |

**Table 31-64:** EPPI\_IMSK Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                           |
|---------|-----------|---------------------------------------------------|
| 3       | LTERRUNDR | Line Track Underflow Error Interrupt Mask.        |
|         |           | 0 Unmask Interrupt                                |
|         |           | 1 Mask Interrupt                                  |
| 2       | LTERROVR  | Line Track Overflow Error Interrupt Mask.         |
|         |           | 0 Unmask Interrupt                                |
|         |           | 1 Mask Interrupt                                  |
| 1       | YFIFOERR  | YFIFO Underflow or Overflow Error Interrupt Mask. |
|         |           | 0 Unmask Interrupt                                |
|         |           | 1 Mask Interrupt                                  |
| 0       | CFIFOERR  | CFIFO Underflow or Overflow Error Interrupt Mask. |
|         |           | 0 Unmask Interrupt                                |
|         |           | 1 Mask Interrupt                                  |

## Clipping Register for ODD (Chroma) Data

The `EPPI_ODDCLIP` register selects the clipping threshold for chroma data, which provides clipping of individual video components.

The high odd and low odd spaces in `EPPI_ODDCLIP` are 16-bits wide and (depending on the `EPPI_CTL.DLEN` selection) only the corresponding video component bits are considered for clipping.

For example, if the EPPI is programmed in 10-bit mode, bits [9:0] and bits 25:16 constitute the clipping thresholds. The higher bits are (in this case) ignored.

Using the this method, 8-, 10-, 12- and 16-bit clipping thresholds can be set.

Note that when the EPPI is programmed in 16-, 20-, or 24-bit mode with `EPPI_CTL.SPLTWRD` set, the luma data gets the clipping threshold levels of `EPPI_EVENCLIP`, and the chroma data gets the clipping threshold levels of `EPPI_ODDCLIP`.

Also note that the EPPI\_EVENCLIP and EPPI\_ODDCLIP registers are ignored when EPPI\_CTL.RGBFMTEN is set.

**EPPI\_ODDCLIP: Clipping Register for ODD (Chroma) Data - R/W**

Reset = 0xffff 0000

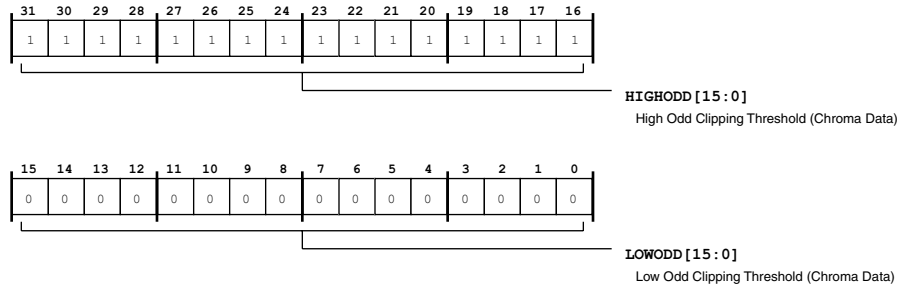


**Figure 31-29:** EPPI\_ODDCLIP Register Diagram

**Table 31-65:** EPPI\_ODDCLIP Register Fields

| Bit No. | Bit Name | Description/Enumeration                    |
|---------|----------|--------------------------------------------|
| 31:16   | HIGHODD  | High Odd Clipping Threshold (Chroma Data). |
| 15:0    | LOWODD   | Low Odd Clipping Threshold (Chroma Data).  |

## Clipping Register for EVEN (Luma) Data

The EPPI\_EVENCLIP register selects the clipping threshold for luma data, which provides clipping of individual video components.

The high even and low even spaces in EPPI\_EVENCLIP are 16-bits wide and (depending on the EPPI\_CTL.DLEN selection) only the corresponding video component bits are considered for clipping.

For example, if the EPPI is programmed in 10-bit mode, bits [9:0] and bits 25:16 constitute the clipping thresholds. The higher bits are (in this case) ignored.

Using the this method, 8-, 10-, 12- and 16-bit clipping thresholds can be set.

Note that when the EPPI is programmed in 16-, 20-, or 24-bit mode with EPPI\_CTL.SPLTWRD set, the luma data gets the clipping threshold levels of EPPI\_EVENCLIP, and the chroma data gets the clipping threshold levels of EPPI\_ODDCLIP.

Also note that the EPPI\_EVENCLIP and EPPI\_ODDCLIP registers are ignored when EPPI\_CTL.RGBFMTEN is set.

EPPI\_EVENCLIP: Clipping Register for EVEN (Luma) Data - R/W

Reset = 0xffff 0000

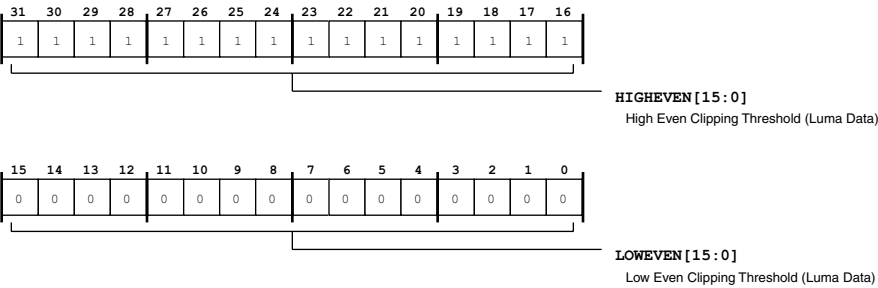


Figure 31-30: EPPI\_EVENCLIP Register Diagram

Table 31-66: EPPI\_EVENCLIP Register Fields

| Bit No. | Bit Name | Description/Enumeration                   |
|---------|----------|-------------------------------------------|
| 31:16   | HIGHEVEN | High Even Clipping Threshold (Luma Data). |
| 15:0    | LOWEVEN  | Low Even Clipping Threshold (Luma Data).  |

Frame Sync 1 Delay Value

The EPPI\_FS1\_DLY selects the delay count (based on the period of the EPPI\_CLK clock) between the first rising edge of EPPI\_CLK after EPPI enabled and the first active edge of the associated Frame Sync when the internal Frame Sync is used.

Note that if EPPI\_FS1\_DLY or EPPI\_FS2\_DLY are programmed with value 0, the EPPI operates as though 0 value is 1, and the first frame sync transition occurs after the completion of one period value of the respective counters.

EPPI\_FS1\_DLY: Frame Sync 1 Delay Value - R/W

Reset = 0x0000 0000

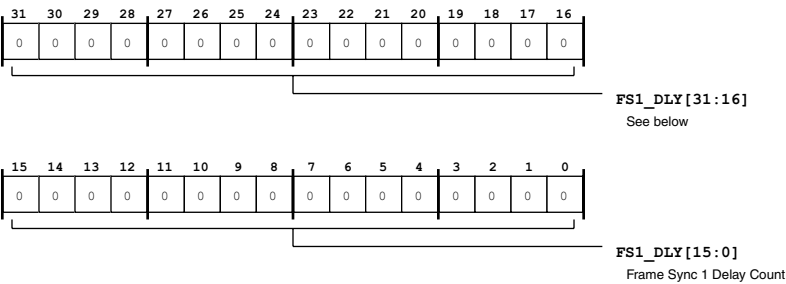


Figure 31-31: EPPI\_FS1\_DLY Register Diagram

**Table 31-67:** EPPI\_FS1\_DLY Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | FS1_DLY  | Frame Sync 1 Delay Count. |

## Frame Sync 2 Delay Value

The EPPI\_FS2\_DLY selects the delay count (based on the period of the EPPI\_CLK clock) between the first rising edge of EPPI\_CLK after EPPI enabled and the first active edge of the associated Frame Sync when the internal Frame Sync is used.

Note that if EPPI\_FS1\_DLY or EPPI\_FS2\_DLY are programmed with value 0, the EPPI operates as though 0 value is 1, and the first frame sync transition occurs after the completion of one period value of the respective counters.

**EPPI\_FS2\_DLY: Frame Sync 2 Delay Value - R/W**

Reset = 0x0000 0000

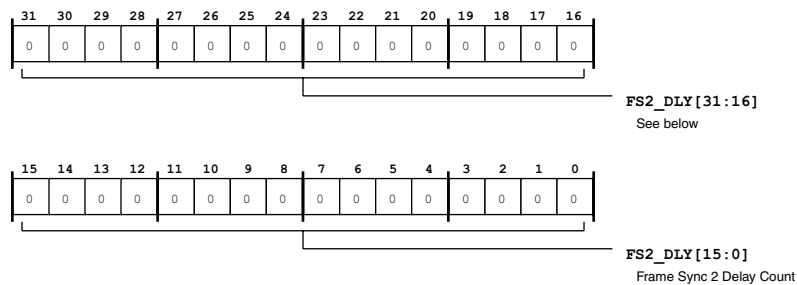


**Figure 31-32:** EPPI\_FS2\_DLY Register Diagram

**Table 31-68:** EPPI\_FS2\_DLY Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:0    | FS2_DLY  | Frame Sync 2 Delay Count. |

## Control Register 2

The EPPI\_CTL2 register HSYNC finish signal generation.

EPPI\_CTL2: Control Register 2 - R/W

Reset = 0x0000 0000

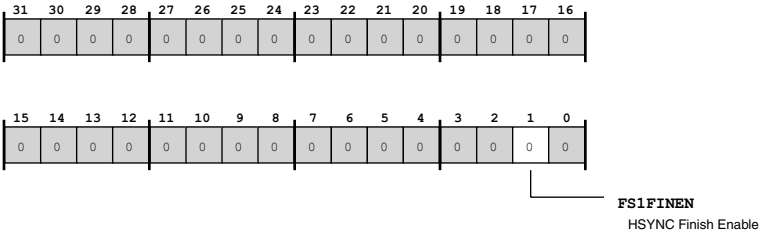


Figure 31-33: EPPI\_CTL2 Register Diagram

Table 31-69: EPPI\_CTL2 Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | FS1FINEN | HSYNC Finish Enable.<br>The EPPI_CTL2.FS1FINEN bit selects whether (if set) the EPPI sends a finish command (010) through the DDE COMMAND line soon after a LINE is received completely or (if cleared) the EPPI sends a finish command (010) through the DDE COMMAND line soon after a FRAME is received completely.<br>Note that the EPPI_CTL.DMAFINEN bit must be set for the EPPI to generate either of the finish commands. |
|         |          | 0<br>Finish sent after frame RX done<br>PPI sends a finish command (010) through the DDE COMMAND line soon after a FRAME is received completely                                                                                                                                                                                                                                                                                  |
|         |          | 1<br>Finish sent after frame/line RX done<br>PPI sends a finish command (010) through the DDE COMMAND line soon after a frame/line is received completely                                                                                                                                                                                                                                                                        |







## 32 Pixel Compositor (PIXC)

The pixel compositor (PIXC) provides data overlay, transparent color, and color space conversion support. This allows for supporting different video outputs including active (TFT) flat-panel digital color/monochrome LCD displays and analog NTSC/PAL. The color space conversion and text/graphic overlay capabilities, along with visual effect controls, such as transparency control, shorten the processing time on an image data stream, reduce power consumption and save system board space by removing the need for external glue logic.

The PIXC is used to combine and format the data streams required by a wide variety of digital LCD panels and NTSC/PAL analog encoders. It provides all the control needed to allow two data streams from two separate data buffers to be combined and converted into appropriate formats for both LCD panels and video output displays. The main image buffer provides the basic background image presented in the data stream. The overlay image buffer allows the user to add foreground text and graphics on top of the main image data stream. This feature is useful for printing additional graphical or textual information on the screen, such as symbols or a menu, while showing the main image in the background.

Overlay is an option and can be enabled or disabled. If it is disabled, the blender/compositor is bypassed and the data stream from the main image buffer goes directly to memory with optional color space conversion.

Transparent color is just a special case of blending, masking off the blend operation on a pixel-by-pixel basis. In other words, the overlay region consists of sub-regions in any particular color convenient to the programmer and then, if the color data for a given overlay pixel matches the specified transparent color, the overlay function is masked for that pixel and its data is taken solely from the main image buffer, which is stored in memory in either YUV 4:2:2 interleaved format or RGB888 format

Regardless of the data format or buffer structure, each color element is 8 bits wide. If overlay is enabled, a graphics/text overlay data buffer is defined in memory. The color space converter can switch positions among any of the three locations; it can be in the image data path, the overlay data path, or after the blender. The exact position of the color space converter depends on the input and output data formats.

Since the end display may be a TV (NTSC/PAL) or an LCD panel, and since the image/overlay input buffers may be in either RGB888 or YUV4:2:2 format, a color space conversion may be needed. The color space conversion is selected according to the input data stream format of the PIXC. A YUV-to-RGB format conversion is necessary if the end display is an LCD and if either of the PIXC input data streams is in YUV 4:2:2 format. Similarly, an RGB-to-YUV format conversion is necessary if the end display is a TV and if either of the PIXC input data streams is in RGB888 format.

If the final display device is an LCD, the output RGB data stream is always packed in RGB 8-bit serial format when transferring back to memory. Similarly, if the final display device is a TV, the YUV data stream is always packed in YUV 4:2:2 interleaved format when transferring back to memory.

## PIXC Features

PIXC features include:

- Hardware-based graphics and text overlays
- YUV 4:2:2, RGB888, RGB666 or RGB565 input data formats
- Programmable color space conversion on the main image or the overlay image data path
- Overlay content transparency ratio control
- Transparent color, specified in the desired color space (RGB or YUV)
- Two DMA input channels and one DMA output channel
- Interface to the pixel pipe

## PIXC Functional Description

The PIXC implements the following main functions.

- Graphics/text overlay (including video overlay for small frame sizes)
- Transparency control (alpha blending) of the overlay pixel data
- Transparent color (chroma keying) of the overlay stream
- Color space conversion for LCD panels or NTSC/PAL displays

## Color Space Converter Output Thresholds

Each PIXC output sample is 8 bits wide, whether it is an R, G, B, Y, U or V component value. Therefore, any output sample must be in the 0 to 255 range. Since all the coefficients are programmable, some of the inputs, when operated upon by the coefficients, may produce an output outside the 0 to 255 range. In such cases, the PIXC clips the output component's value to 0 or 255.

## YUV Conversion Modes

When the color space converter operates between two color spaces, it requires all components of each pixel to be present in the data stream.

Therefore, the PIXC internally up samples the YUV 4:2:2 data stream before a YUV-to-RGB conversion and similarly down samples the YUV 4:4:4 data stream after a RGB-to-YUV conversion. The re-sampling

always takes place between YUV 4:2:2 and YUV 4:4:4 formats, but a certain flexibility is provided with regard to how the re-sampling is done by the PIXC in each case.

- Up sampling

A YUV 4:2:2-to-YUV 4:4:4 conversion can be performed either by averaging or by duplicating the pixel components. The UDSMOD bit in the PIXC\_CTL register specifies the up sampling mode. The default setting of this bit is 0, which corresponds to duplication of the chroma components (Us and Vs) from the odd pixels to the even pixels:

|                       |                                     |
|-----------------------|-------------------------------------|
| YUV 4:2:2 input:      | V1Y1, U1Y2, V3Y3, U3Y4, ...         |
| YUV 4:4:4 conversion: | Y1U1V1, Y2U1V1, Y3U3V3, Y4U3V3, ... |

Setting the PIXC\_CTL.UDSMOD bit to 1 enables the averaging of the chroma components of the preceding and succeeding pixels to obtain the intermediate chroma value. In other words, two consecutive odd-numbered pixels' chroma components are averaged to obtain the intermediate even-numbered pixel's chroma components:

|                       |                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| YUV 4:2:2 input:      | V1Y1, U1Y2, V3Y3, U3Y4, ...                                                                                       |
| YUV 4:4:4 conversion: | Y1U1V1, Y2U2V2 [ $U2=(U1+U3)/2$ ,<br>$V2=(V1+V3)/2$ ], Y3U3V3, Y4U4V4<br>[ $U4=(U3+U5)/2$ , $V4=(V3+V5)/2$ ], ... |

If the sum of the preceding and succeeding pixels' U/V components is an odd number, the average is rounded down (truncated to an integer value).

Since the last pixel on a line is always an even-numbered pixel, the last odd pixel value on that line is used as the last even pixel value during up sampling.

- Down Sampling

A YUV 4:4:4-to-YUV 4:2:2 conversion can be performed either by averaging or by dropping the pixel components. The PIXC\_CTL.UDSMOD bit also governs the down sampling mode. Setting the PIXC\_CTL.UDSMOD bit to 0 (default) enables the dropping of the chroma components of the even numbered pixels:

|                       |                                     |
|-----------------------|-------------------------------------|
| YUV 4:4:4 input:      | Y1U1V1, Y2U2V2, Y3U3V3, Y4U4V4, ... |
| YUV 4:2:2 conversion: | V1Y1, U1Y2, V3Y3, U3Y4, ...         |

|                       |                                     |
|-----------------------|-------------------------------------|
| YUV 4:4:4 input:      | Y1U1V1, Y2U2V2, Y3U3V3, Y4U4V4, ... |
| YUV 4:2:2 conversion: | V12Y1, U12Y2, V34Y3, U34Y4, ...     |
|                       | $[U12=(U1+U2)/2, U34=(U3+U4)/2]$    |
|                       | $[V12=(V1+V2)/2, V34=(V3+V4)/2]$    |

## PIXC Actions

The table below lists the PIXC actions that take place based on any possible combination of image, overlay, and output data formats.

**Table 32-1:** PIXC Actions

| YUV | No Overlay | RGB | US followed by CSC                                      |
|-----|------------|-----|---------------------------------------------------------|
| RGB | No Overlay | YUV | CSC followed by DS                                      |
| YUV | YUV        | YUV | US in both paths followed by DS before output           |
| YUV | RGB        | RGB | US in image path, CSC in image path                     |
| YUV | YUV        | RGB | US in both paths, followed by CSC                       |
| YUV | RGB        | YUV | CSC in overlay path, US in image path, DS before output |
| RGB | YUV        | YUV | CSC in image path, US in overlay path, DS before output |
| RGB | YUV        | RGB | US in overlay path, CSC in overlay path                 |
| RGB | RGB        | YUV | CSC followed by DS                                      |
| RGB | RGB        | RGB | No CSC, No US, No DS                                    |

- CSC = Color Space Conversion
- US = Up sampling
- DS = Down Sampling
- YUV = YUV 4:2:2 format

- RGB = RGB888 format

## Recommendation

For best results, the overlay should start on an odd-numbered pixel so that the U and V components of the image and the overlay are aligned. Otherwise artifacts may occur in the combined image.

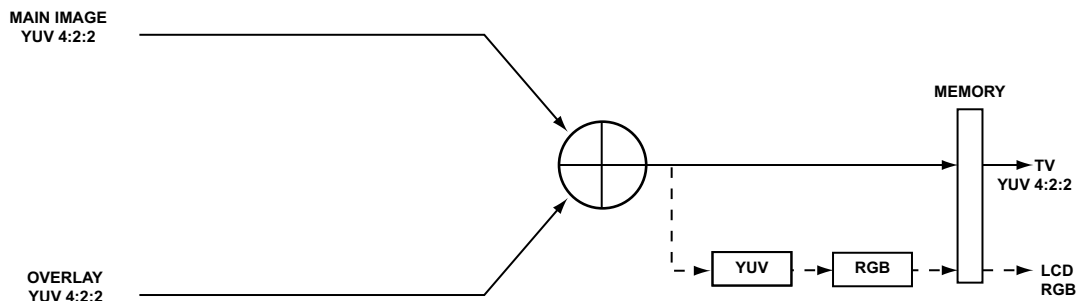
When both the image and the overlay are in YUV 4:2:2 format and the output is also in YUV 4:2:2 format, the duplicating-dropping mode (PIXC\_CTL.UDSMOD) is used to prevent a low-pass filtering effect on the images

## Special Use Cases

There are ways by which the PIXC can be made to operate on certain data formats that it does not support in any standard modes. For example, YUV 4:4:4 is similar to RGB 888 with respect to the number of pixels per 32-bit DMA word. So the PIXC can be configured to work with the YUV 4:4:4 data format by intelligently programming the IFRMT, OVFRMT, and OUTFRMT bit fields in the PIXC\_CTL register and the color space conversion coefficients.

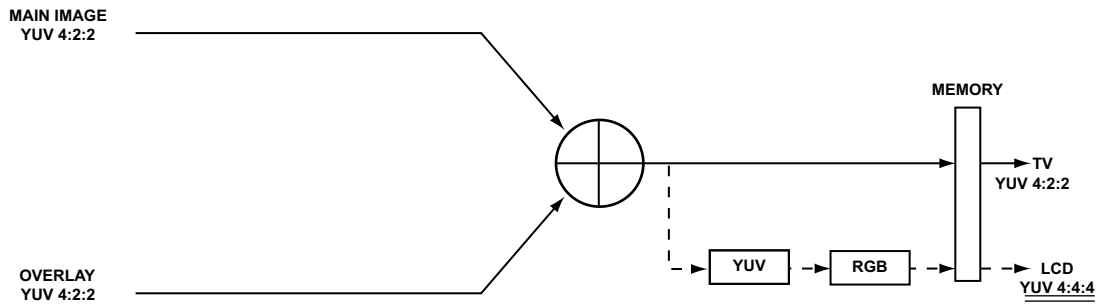
These special usage cases are shown in the figures below.

### Example 1 - Currently Defined Mode

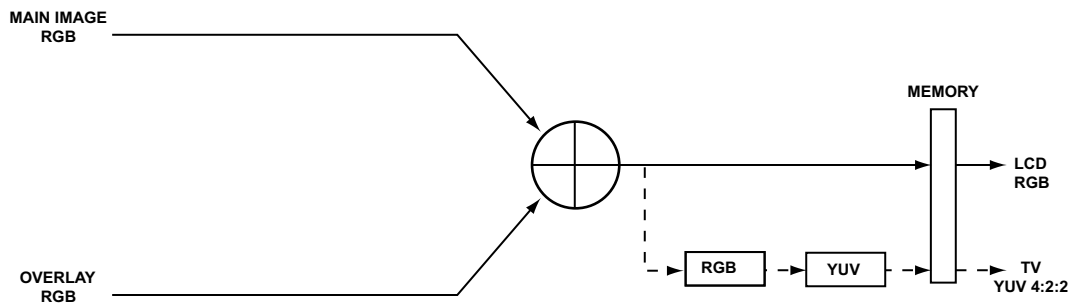


### Example 1 - Special Usage of this Mode

- IFRMT = YUV
- OVFRMT = YUV
- OUTFRMT = RGB

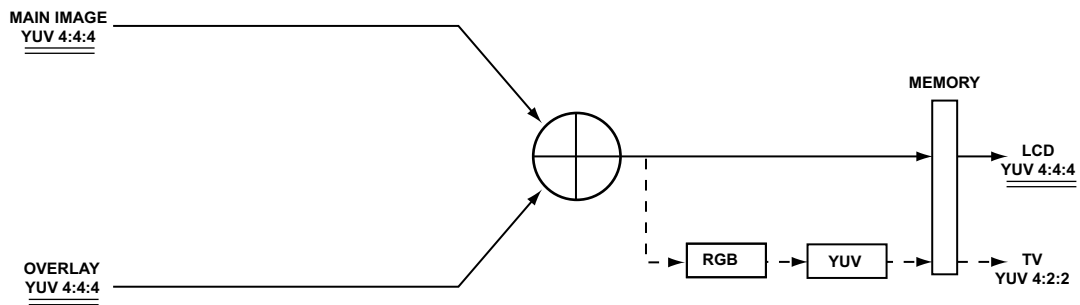


## Example 2 - Currently Defined Mode

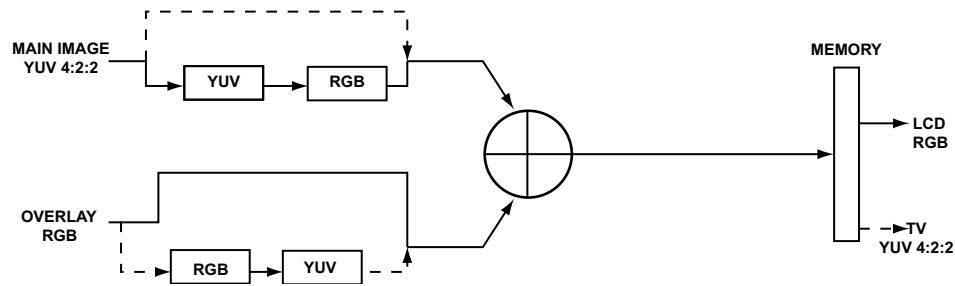


## Example 2 - Special Usage of this Mode

In the special usage of this mode, YUV 4:4:4 input produces a blended YUV 4:2:2 or YUV 4:4:4 data stream. A CSC matrix with coefficients of 1 is needed.

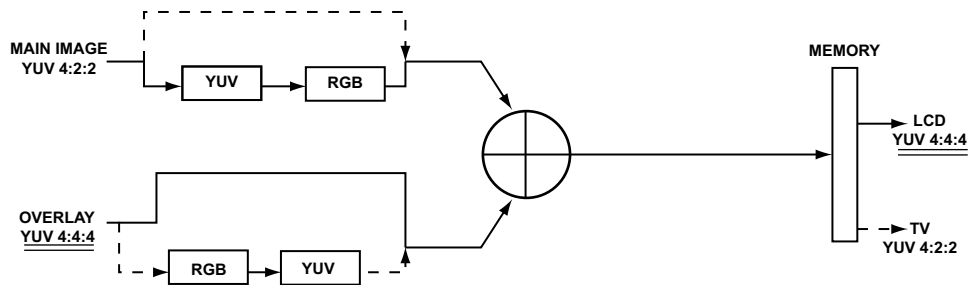


### Example 3 - Currently Defined Mode

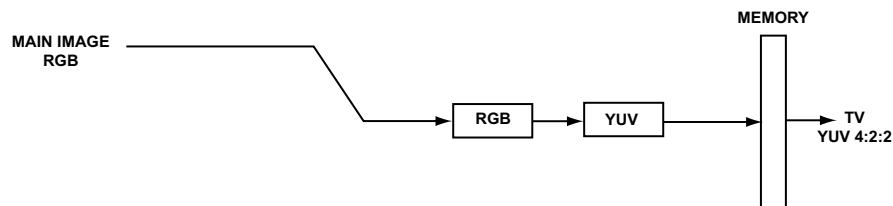


### Example 3 - Special Usage of this Mode

In the special usage of this mode, a YUV 4:4:4 input stream and a YUV 4:2:2 input stream can be blended to produce either a YUV 4:4:4 or a YUV 4:2:2 output stream.



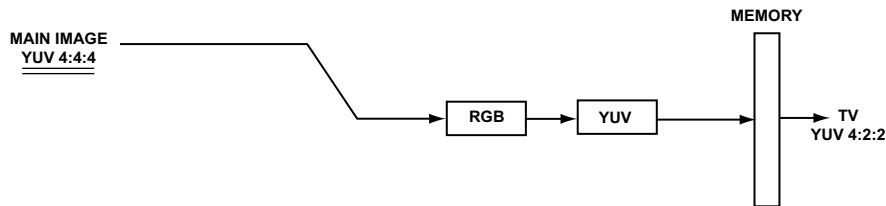
### Example 4 - Currently Defined Mode



### Example 4 - Special Usage of this Mode

- PIXC\_EN = 1
- OVR\_A\_EN = 0
- OVR\_B\_EN = 0

- IMG\_FORM = RGB
- OUT\_FORM = YUV
- All CSC coefficients = 1



## PIXC Register List

The pixel compositor (PIXC) provides data overlay, transparent color, and color space conversion support for active (TFT) flat-panel digital color/monochrome LCD displays or analog NTSC/PAL video output. A set of registers govern PIXC operations. For more information on PIXC functionality, see the PIXC register descriptions. Programmers should avoid writing to any of the PIXC registers when the module is enabled. Writing to the PIXC registers during the module enabled state can lead to unpredictable behavior of the PIXC. All registers can be read when the PIXC is in the enabled state, and this does not cause any change of status in the PIXC, but register writes should happen only when the PIXC is disabled, or stalled by an interrupt condition.

**Table 32-2:** PIXC Module Registers

| Name          | Description                           |
|---------------|---------------------------------------|
| PIXC_CTL      | Control Register                      |
| PIXC_PPL      | Pixels Per Line Register              |
| PIXC_LPF      | Line Per Frame Register               |
| PIXC_HSTART_A | Overlay A Horizontal Start Register   |
| PIXC_HEND_A   | Overlay A Horizontal End Register     |
| PIXC_VSTART_A | Overlay A Vertical Start Register     |
| PIXC_VEND_A   | Overlay A Vertical End Register       |
| PIXC_TRANSP_A | Overlay A Transparency Ratio Register |
| PIXC_HSTART_B | Overlay B Horizontal Start Register   |
| PIXC_HEND_B   | Overlay B Horizontal End Register     |



**Table 32-2:** PIXC Module Registers (Continued)

| Name          | Description                           |
|---------------|---------------------------------------|
| PIXC_VSTART_B | Overlay B Vertical Start Register     |
| PIXC_VEND_B   | Overlay B Vertical End Register       |
| PIXC_TRANSP_B | Overlay B Transparency Ratio Register |
| PIXC_IRQSTAT  | Interrupt Status Register             |
| PIXC_CONRY    | RY Conversion Component Register      |
| PIXC_CONGU    | GU Conversion Component Register      |
| PIXC_CONBV    | BV Conversion Component Register      |
| PIXC_CCBIAS   | Conversion Bias Register              |
| PIXC_TC       | Transparency Color Register           |
| PIXC_REVID    | Revision Id                           |

## ADSP-BF60x PIXC Interrupt List

**Table 32-3:** ADSP-BF60x PIXC Interrupt List

| Description         | ID  | DMA Channel | Sensitivity |
|---------------------|-----|-------------|-------------|
| PIXC0 Channel 0 DMA | 107 | 35          | LEVEL       |
| PIXC0 Channel 1 DMA | 108 | 36          | LEVEL       |
| PIXC0 Channel 2 DMA | 109 | 37          | LEVEL       |
| PIXC0 Status        | 110 |             | LEVEL       |

## ADSP-BF60x PIXC Trigger List

**Table 32-4:** Masters ADSP-BF60x PIXC Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| PIXC0 Channel 0 DMA | 57 | PULSE/EDGE  |
| PIXC0 Channel 1 DMA | 58 | PULSE/EDGE  |
| PIXC0 Channel 2 DMA | 59 | PULSE/EDGE  |
|                     |    |             |

**Table 32-5:** Slaves ADSP-BF60x PIXC Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| PIXC0 Channel 0 DMA | 57 |             |
| PIXC0 Channel 1 DMA | 58 |             |
| PIXC0 Channel 2 DMA | 59 |             |
|                     |    |             |

## PIXC Definitions

To make the best use of the PIXC, it is useful to understand the following terms.

### NTSC/PAL

The two most common video standards used. NTSC is the video system or standard used in North America and most of South America. In NTSC, 30 frames are transmitted each second. Each frame is made

up of 525 individual scan lines. PAL is the predominant video system or standard mostly used overseas. In PAL, 25 frames are transmitted each second. Each frame is made up of 625 individual scan lines.

### RGB888 format

RGB is a color space where pixels are defined by three color values; one red (R), one green (G) and one blue (B). The suffix signifies the bit widths for these color components. In this case, RGB888 means that each red, green and blue value is 8-bits each.

### YUV 4:2:2 interleaved format

YUV is a color space where pixels are defined by a luminance (Y) component and chrominance (UV) components. The suffix signifies how the chrominance components have been decimated as well as formatting. In this case, the YUV422 format has the chrominance decimated by two, meaning only half of each chrominance component are available. Typical YUV422 formatting interleaves the luminance and chrominance such as U1Y1V1Y2U2Y3V2Y4.

### Data Overlay

Process of blending or replacing a pixel from an image with another pixel.

## PIXC Block Diagram

A top-level micro architecture diagram of the PIXC appears in the following figure. As shown in the figure, the PIXC uses three DMA channels: one for the image data, one for the overlay data and one for storing the results back to memory. Frame C (output frame) can also be fed back to the PIXC for multiple stages of processing, taking the place of frame A (main input image) when this happens. The main input image and output frame can alternatively be piped from or to another peripheral via the Video Subsystem, respectively.

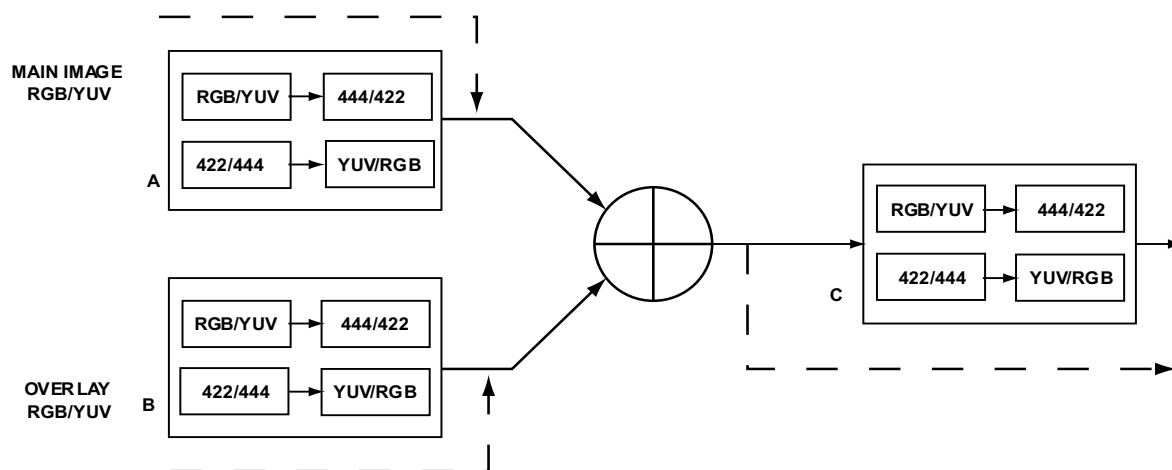


**Figure 32-1:** Pixel Compositor Top-Level Diagram

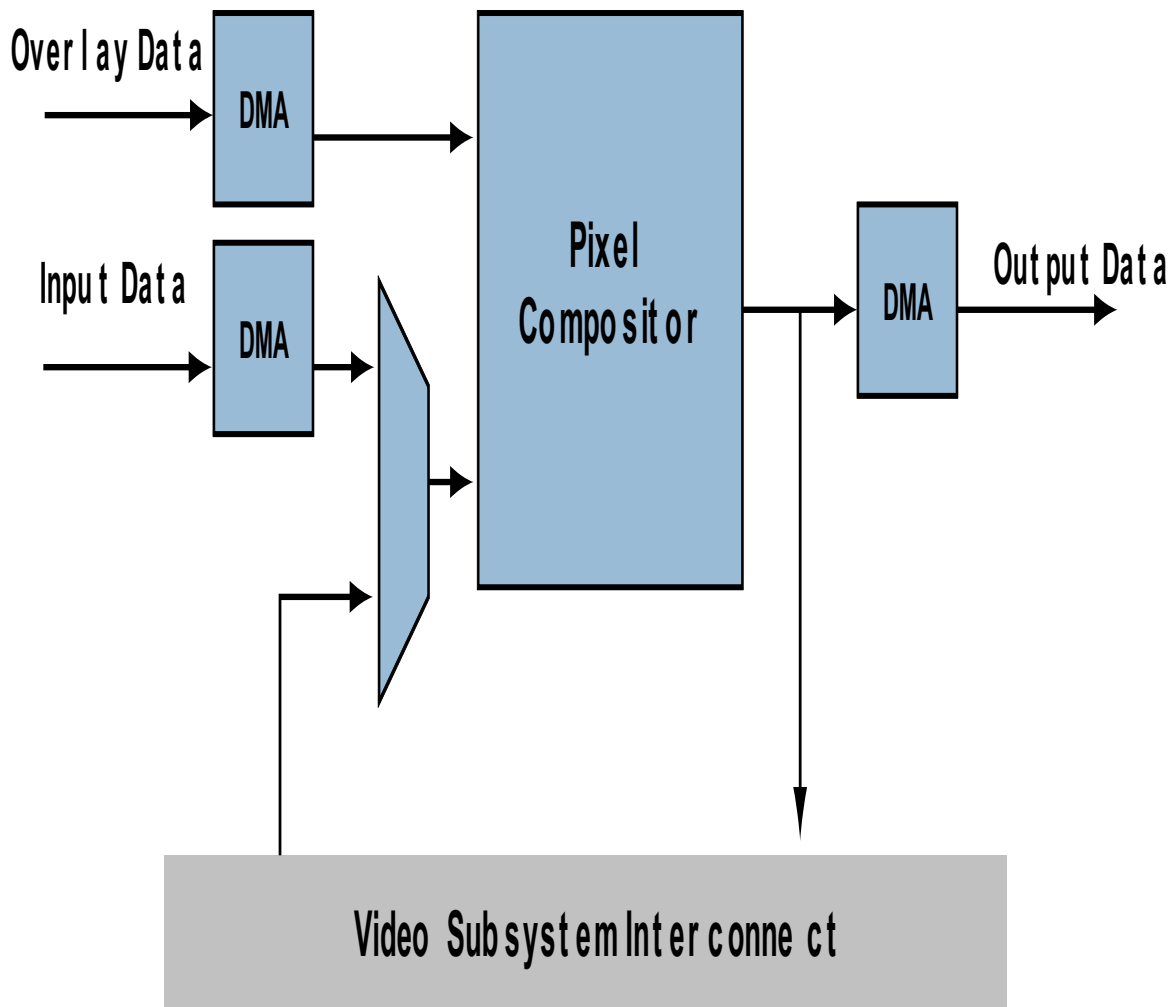


**Figure 32-2:** PIXC Functional Block Diagram

## PIXC Architectural Concepts

The PIXC data interface can be configured to obtain the input main image from memory via DMA or piped from a different peripheral configured by the Video Subsystem. The output result image can also be sent back to memory via DMA or piped to a peripheral. Also, the output can be fed back into the PIXC for multiple stages of processing. The overlay image data must be brought in from memory via DMA.

### Pixel Pipe (PxP) Interface

The PIXC supports a pixel pipe (PxP) interface at the input as well as the output sides. The input side is multiplexed with the DMA channel carrying the input image data from memory. The video subsystem connectivity determines whether the PIXC image data source is connected to the IFIFO or the Pixel Pipe. The output composite image/color converted image can be either transferred to memory via DMA or to the PVP/EPPI via the PxP. This is also determined by the video subsystem connectivity.

If the PIXC image input is programmed to connect to the PxP bus then:

1. The PIXC expects only complete frames from the PxP. Each line should contain the exact number of pixels as programmed in the `PIXC_PPL` register and each frame should contain as many lines as programmed in the `PIXC_LPF` register. The EPPI is the only source of data when the PIXC input connects to the PxP. It is ensured by the EPPI that once a frame starts, no further frame sync is passed on the PxP bus until the frame is completed.
2. Internal signaling is used to start the pixel processing in the PIXC. The PPL counter increments at pixel valid. The LPF counter is incremented only when the PPL counter rolls over and resets to one when the max LPF count is reached (as programmed in the `PIXC_LPF` register).

#### PIXC Start Synchronization in PxP Input Mode

When the PIXC is enabled after configuring it to take the image data from the PxP bus, the PIXC internal logic synchronizes itself to the incoming PxP frame sync. The PIXC ignores all the incoming data until it gets this PxP frame sync from the input PxP bus. After the internal frame sync is recognized by the PIXC, its engine starts processing the incoming data. This removes any restriction of the relative enable times of the PIXC and the EPPI. Either could be enabled first. If the EPPI is enabled first and the PIXC is enabled later, the PIXC would start its frame processing only at the start of the next frame after PIXC is enabled.

## DMA Interface

The PIXC has a native DMA controller with three channels. Two are used for the input; one for transferring the input image to the PIXC from memory and one for transferring the overlay image to the PIXC from memory. One channel is used for the output, transferring the blended data to memory.

The two input DMA channels with 32-bit bus widths take the image data and overlay graphics/text data from their buffers into two separate FIFO's where the data is then unpacked. Each of these FIFO buffers is 32 bits wide and contains 8 entries.

In the blender, (8-bit) pixel elements from two buffers are mixed together. One dedicated DMA channel transfers the combined pixel data back to memory. Each channel is a DDE instance. A local arbiter arbitrates between the three channels. This arbitrated request will be forwarded to the system crossbar. The PIXC only has one connection to the system crossbar.

When using the DMA to interface to the PIXC, at least two DMA channels should be enabled and configured appropriately; The image DMA channel and the output DMA channel. Furthermore, when the overlay function of PIXC is enabled, the overlay DMA channel should be enabled and configured as well.

## PIXC Data Overlay

Overlay is an optional function, so it can be enabled or disabled. If it is disabled, all overlay functionality is bypassed, and a single data stream from the main image data buffer goes directly to the image output buffer, after an optional format conversion. If it is enabled, the blender combines the pixel data from the two image input buffers.

The overlay image is located in a user-defined rectangle within the main image and, in most cases, the overlay image is smaller than the main image. The figure below illustrates an example of the main and overlay image regions on a screen, where a foreground triangle overlay sits on top of the main image in the background. Although the figure does not show this explicitly, (H-Start, V-Start) can equal (0,0).

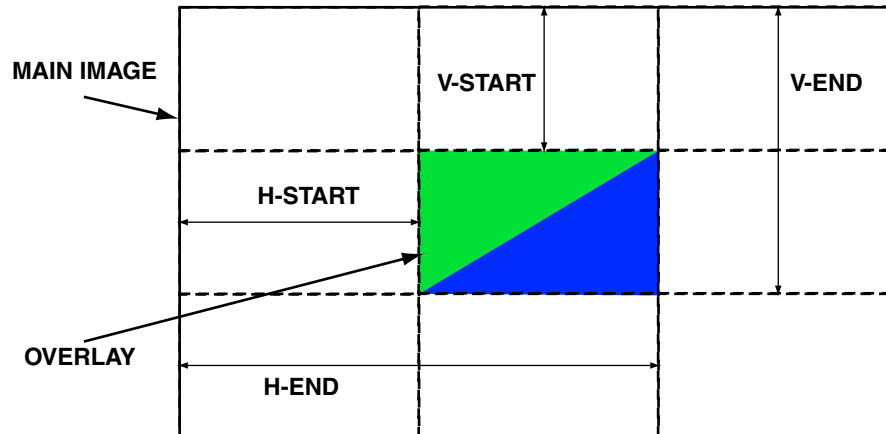


**Figure 32-3:** Main Image and Overlay Image Region

In certain situations, it may be beneficial to only DMA the region of the main image that is affected by the overlay instead of bringing in the entire main image. For example, in a situation where the intent is to overlay an image over the main image and to store the result back over the main image, one could setup a 2D-DMA to only bring in the area of the main image that is affected by the overlay. This may reduce the amount of DMA activity thus potentially improving system performance.

There is a set of an additional registers that can be used to specify a second overlay region, so that two separate overlay blocks can be defined simultaneously. Furthermore, either or both of these overlay coordinate register sets can be enabled or disabled at one time, since separate enable bits in the PIXC\_CTL register (OVAEN and OVBEN) exist in the PIXC control register for each of the overlay register sets.

If there are more than two overlay blocks needed in a given application, the two sets of overlay registers must be managed by the user to perform the additional overlays. This can be done using an interrupt service routine, where the interrupt from the PIXC is used to re-program the overlay coordinate registers.

The PIXC can generate an interrupt under two conditions: at the end of the last valid overlay and at the end of a frame.

Either of these interrupts can be enabled or disabled. However, the PIXC only has one interrupt line output, so it raises an interrupt (under the appropriate condition) when either of these two interrupts is triggered. If both interrupts are enabled, the interrupt status register of the PIXC indicates which of the two conditions caused the interrupt to occur. Once the PIXC generates an interrupt, it stalls the pixel processing until software (ISR) clears the interrupt. However, the FIFOs do not stall and keep filling up even when the PIXC is in a stalled state. Both interrupts can be cleared by writing a 1 to the respective interrupt status bits.

After each interrupt (whether it is a last-valid-overlay interrupt or an end-of-frame interrupt), the PIXC restarts processing with coordinate register set A. In other words, at the time of clearing the interrupt:

- If coordinate set A is enabled (`PIXC_CTL.OVAEN = 1`), the PIXC assumes that the first incoming data over the bus is to be overlaid on the area specified in coordinate set A.
- If coordinate set A is disabled (`PIXC_CTL.OVAEN = 0`), and coordinate set B is enabled (`PIXC_CTL.OVBEN = 1`), the PIXC assumes that the first incoming data over the data bus is to be overlaid on the area specified in coordinate set B.
- If both coordinate sets are disabled, the PIXC flushes the overlay FIFO and make no more data requests on the overlay DMA channel.

**NOTE:** The overlay enable bits `PIXC_CTL.OVAEN` and `PIXC_CTL.OVBEN` should only be changed inside the interrupt service routines of the PIXC interrupts, or when the overlay block is disabled.

Note that the module enable bit (`PIXC_CTL.EN`) is the root enable for the PIXC. Both the `OVAEN` and `OVBEN` bits in the `PIXC_CTL` register are gated with `PIXC_CTL.EN` bit, so if `PIXC_CTL.EN = 0`, the individual overlay enable bits have no effect, and the module remains disabled. When `PIXC_CTL.EN = 0`, both the image and overlay FIFOs are flushed and no more DMA requests are made on either of the DMA channels.

Once the DMAs are enabled, the PIXC keeps track of the current pixel being displayed from the main image data by reading from two user-programmable registers: `PIXC_PPL`, which stores the number of pixels per line, and `PIXC_LPF`, which stores the number of lines per frame of the display device.

When the pixel count reaches the top left corner (H-Start, V-Start) of overlay data, the PIXC starts the overlay. When the pixel count reaches the top right corner (H-End, V-Start) of overlay data, the PIXC stops the overlay. It starts again at the next line at (H-Start, V-Start+1) and stop at (H-End, V-Start + 1), and so on until the entire overlay frame is processed.

**NOTE:** Internally, the start of the overlay DMA would have been preempted by the PIXC before the actual processing of the first overlay pixel, and DMA data would have been requested until the overlay FIFO were full. Similarly, the overlay DMA does not stop at the end of a line. The overlay FIFO continues to be filled with DMA data, even when the current pixel is not an overlay pixel, but the supply of overlay pixels from the overlay FIFO is simply halted.

The PIXC decides whether or not to perform overlay mixing for the current pixel by using the various PIXC register values. Therefore:

- The `PIXC_PPL` and `PIXC_LPF` must be programmed correctly (and cannot be 0).
- The `HSTART` and `HEND` must be less than or equal to `PIXC_PPL`.
- The `VSTART` and `VEND` must be less than or equal to `PIXC_LPF`.

The user can define multiple rectangular regions covering several separate overlays, using the same number of DMA descriptors, where each DMA descriptor corresponds to an overlay region.

Multiple overlay regions are split into two cases:

- Overlay regions with no horizontal overlap – Here software can maintain separate areas in memory for both overlay regions, with separate H-Start, V-Start, H-End, and V-End coordinates for each region. After the first overlay is completed, the DMA chain pointer can load the next overlay parameters (index, count, and modifier) to the DMA registers of the corresponding DMA channel.

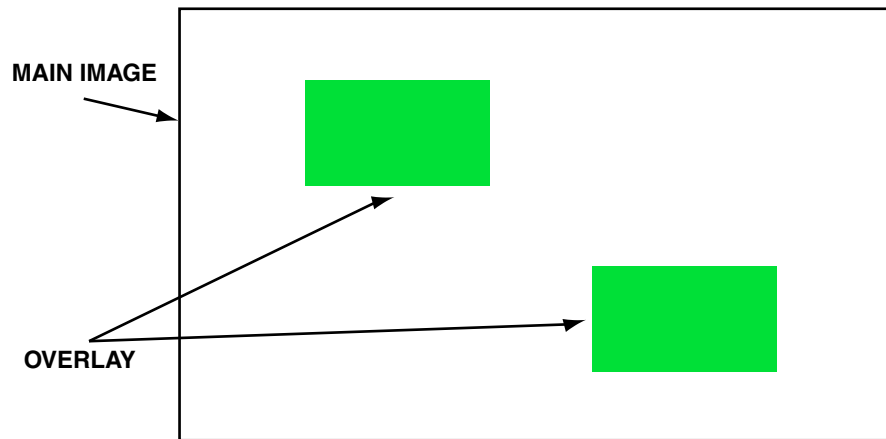


**Figure 32-4:** Overlay Regions With No Horizontal Overlap

- Overlay regions with horizontal overlap – In these cases software has to maintain a combined overlay region in memory. This includes some in-between area where there is no overlay. This region of memory has to be filled with the transparent color value (explained below). The H-Start, V-Start, H-End and V-End coordinates contain the values of the combined overlay region.

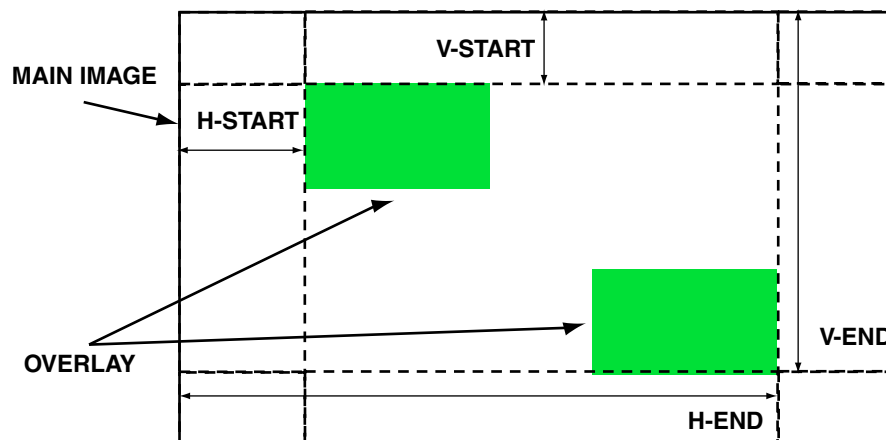


**Figure 32-5:** Overlay Regions With Horizontal Overlap

## PIXC Transparency Control

When the overlay function is enabled, each overlay pixel is combined with each main image pixel to generate the displayed output pixel to be displayed. Each pixel combination is controlled by a transparency



ratio value alpha ( $\alpha$ ), a 4-bit value that determines the proportion of overlay and main image that contribute to the output pixel. The pixel combination algorithm can be expressed as:

$$C = \frac{B(\alpha + 1)}{16} + \frac{A(15 - \alpha)}{16}$$

The table lists the multiplying factors for various  $\alpha$  values.

- A = 8-bit pixel data in main frame buffer (“background”)
- B = 8-bit pixel data in overlay buffer (“foreground”)
- C = 8-bit combined pixel data
- $\alpha$  = Transparency ratio code, which is a 4-bit value present in a memory-mapped register

**Table 32-6:** Multiplying Factors for Various  $\alpha$  Values

| $\alpha$ | Overlay Multiplying Factor | Image Multiplying Factor |
|----------|----------------------------|--------------------------|
| 0        | 1/16                       | 15/16                    |
| 1        | 2/16                       | 14/16                    |
| 2        | 3/16                       | 13/16                    |
| 3        | 4/16                       | 12/16                    |
| 4        | 5/16                       | 11/16                    |
| 5        | 6/16                       | 10/16                    |
| 6        | 7/16                       | 9/16                     |
| 7        | 8/16                       | 8/16                     |
| 8        | 9/16                       | 7/16                     |
| 9        | 10/16                      | 6/16                     |
| 10       | 11/16                      | 5/16                     |
| 11       | 12/16                      | 4/16                     |
| 12       | 13/16                      | 3/16                     |
| 13       | 14/16                      | 2/16                     |

**NOTE:** Passing the image alone can be achieved by disabling the overlay function.

Rounding is performed at the output of the blender, which rounds the combined pixel data to the nearest integer value.

## PIXC Transparency Color

A transparent color is a specific color that is removed from one image to reveal another “behind” it. This technique is also referred to as chroma keying. The principal subject is photographed or filmed against a background having a single color, usually in the blue or green spectrum. When the phase of the chroma signal corresponds to the pre-programmed state associated with the background color(s) behind the principal subject, the signal from the alternate background (which in this case comes from the main image channel) is inserted in the composite signal and presented at the output. When the phase of the chroma signal deviates from that associated with the background color(s) behind the principal subject, the picture data associated with the principal subject (in this case, the overlay image) is presented at the output. The figure below illustrates this concept.

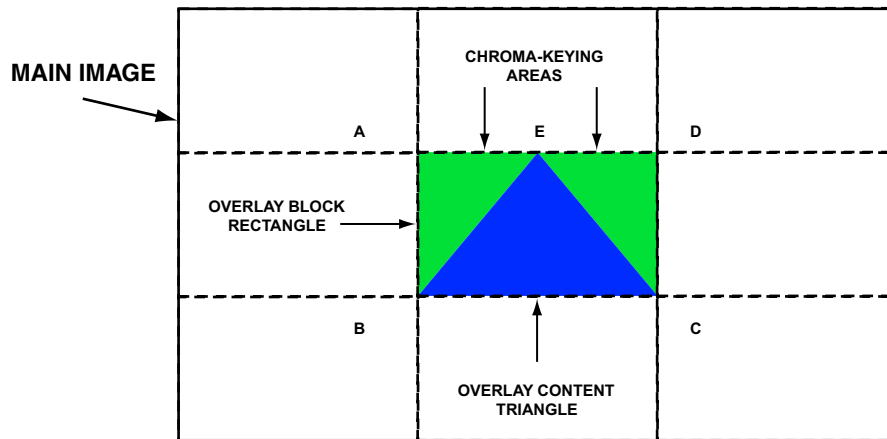


**Figure 32-6:** Transparent Color (Chroma Keying)

In order to display the main image in the two triangle areas  $\triangle ABE$  and  $\triangle CDE$  in overlay block ABCD, the data in the overlay buffer corresponding to the pixels in the triangle areas  $\triangle ABE$  and  $\triangle CDE$  must hold a specific value, called the transparent color.

The PIXC provides a 24-bit MMR (storing three 8-bit color components) for each of the two overlay blocks to designate a particular RGB or YUV value as the transparent color. The transparent color must be in the same format (YUV 4:2:2 or RGB888) as the overlay data, regardless of whether or not a color space conversion is present in the overlay data path. The PIXC then compares each input pixel value on the overlay channel with this transparent color. If there is a match, the overlay pixel at this location is ignored by the blender, and the main image pixel at that location is assigned 100% weight.

**NOTE:** If YUV 4:2:2 is the overlay channel input data format, artifacts may occur at the edge of the transparent color region. In this case, it is preferable to set the `PIXC_CLT.UDSMOD` bit to 0 (duplicating-dropping mode), in order to get better control of the U and V components at the edge of the transparent color region.

## PIXC Color Space Conversion

Depending on the input data format and display device used, there may be a color space conversion performed on the data stream of the PIXC. If the input data is in YUV format, a YUV-to-RGB conversion can be performed for output to an LCD panel. If the input data is in RGB format, a RGB-to-YUV conversion can be performed for output to NTSC/PAL displays. The color space conversion may happen on any of the three paths (for example, the main image data path, the overlay image data path, or the combined data path). Register bits are used to specify the input, overlay and output formats. The color space converter block controls a number of cases of operation:

- Both the image and the overlay data are in the same format.
- The image and the overlay data are in different formats.
- Color space conversion only

Each case is described below, along with several special usage cases. Note that various scenarios may be shown in the same figure based on the output device chosen, though only a single output destination is supported at one time.

### Case 1 - Image and Overlay in the Same Format

Both input data streams (main image and overlay) are in the same format, either YUV 4:2:2 or RGB888, so a color space conversion may be performed after alpha blending, depending on the output type.

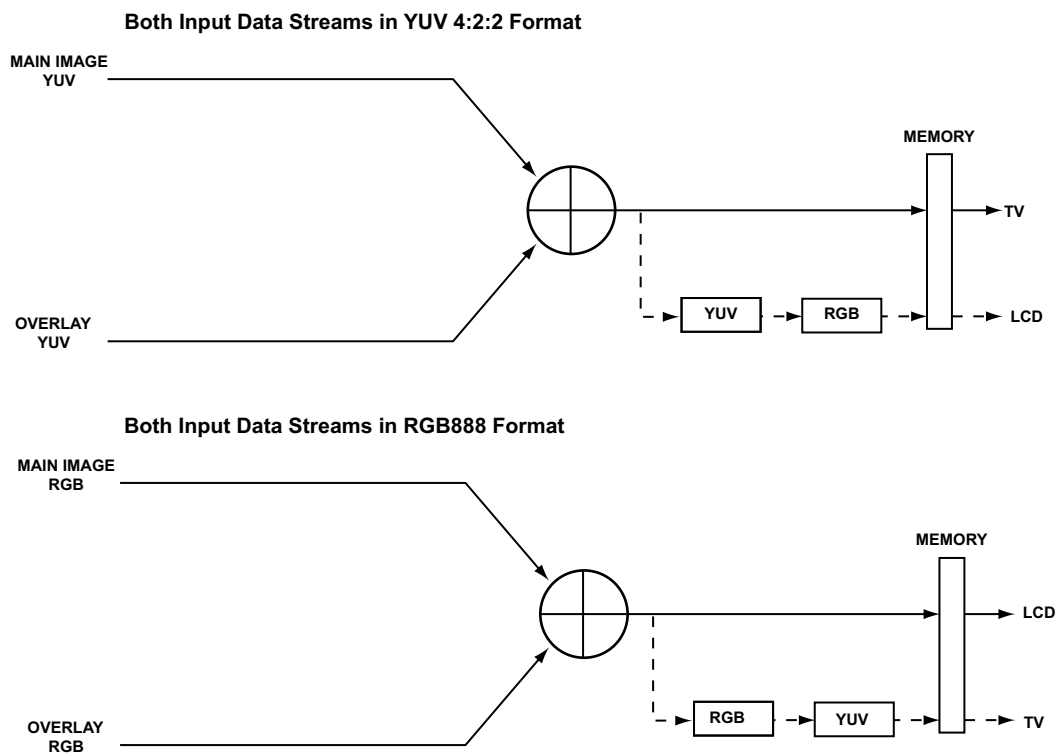


**Figure 32-7:** Image and Overlay in the Same Format

## Case 2 - Image and Overlay in Different Formats

In this case, the two input data streams are not in the same format. The PIXC has to perform a color space conversion on either the main input stream or the overlay input stream (depending on the required output format) before alpha blending can take place.

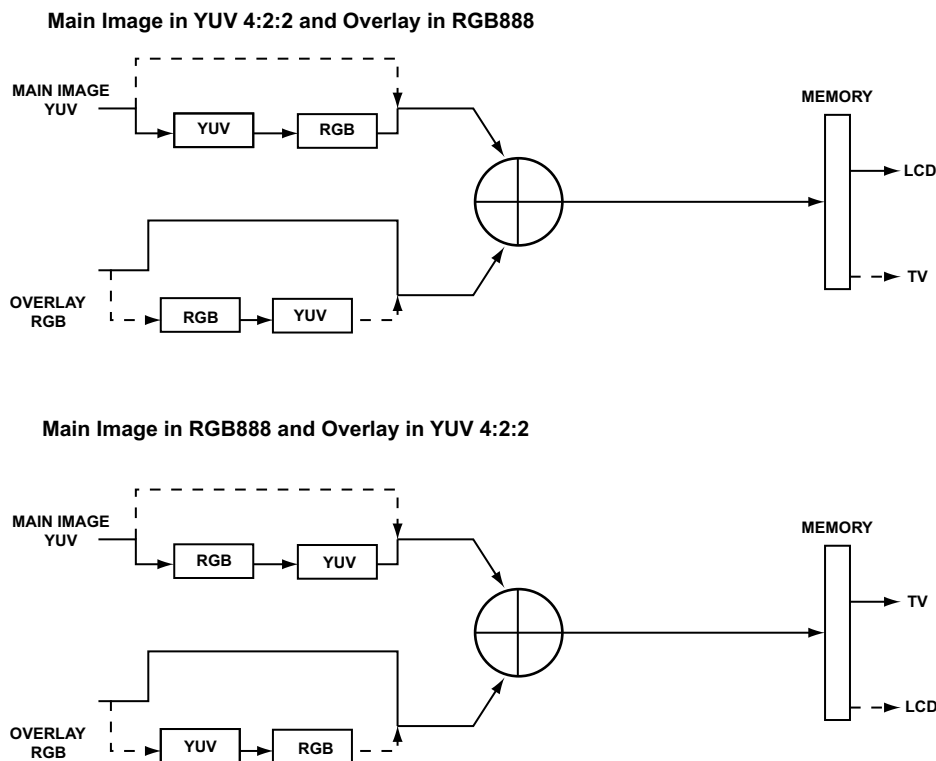


**Figure 32-8:** Image and Overlay in Different Formats

## Case 3 - Color Space Conversion Only

In this case, there is no overlay blending. The main image is brought into the PIXC, the color space converted, and then sent back to memory.

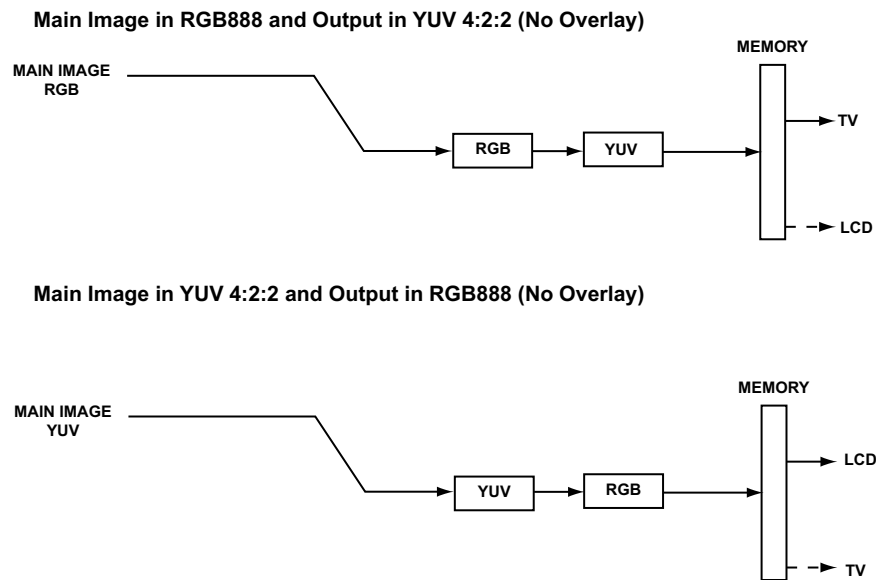


**Figure 32-9:** Color Space Conversion Only

For this mode, the register settings are: `PIXC_CTL.EN = 1`, `PIXC_CTL.OVAEN = 0` and `PIXC_CTL.OVBEN = 0`.

## PIXC Operating Modes

The color space converter block provides conversion operations in the following situations.

- When both the image and the overlay data are in the same format.
- When the image and the overlay data are in different formats.
- For color space conversion only

These situations are described in detail in the following sections. Note that various scenarios may be shown in the same figure based on the output device chosen, though only a single output destination is supported at one time.

### PIXC Mode Case 1 - Image/Overlay in the Same Format

Both input data streams (main image and overlay) are in the same format, either YUV 4:2:2 or RGB888, so a color space conversion may be performed after alpha blending, depending on the output type. See figures below.

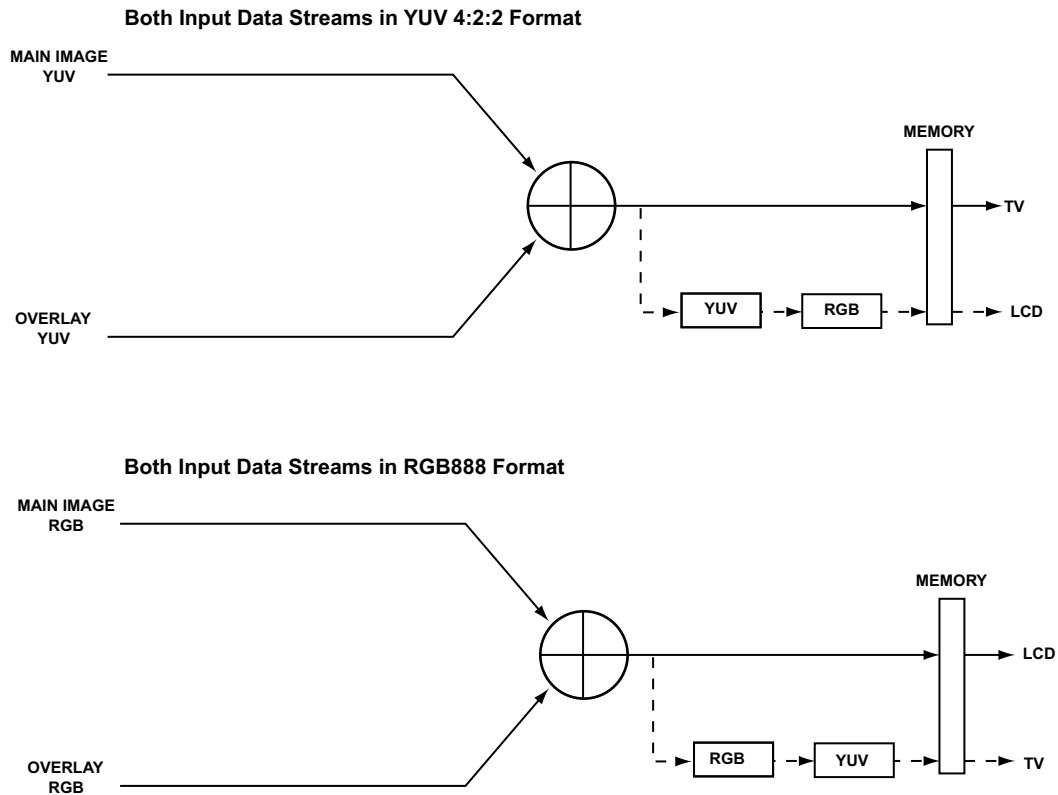


**Figure 32-10:** Image and Overlay in the Same Format

## PIXC Mode Case 2 - Image/Overlay in Different Formats

In this case, the two input data streams are not in the same format. The PIXC has to perform a color space conversion on either the main input stream or the overlay input stream (depending on the required output format) before alpha blending can take place. See figures below.

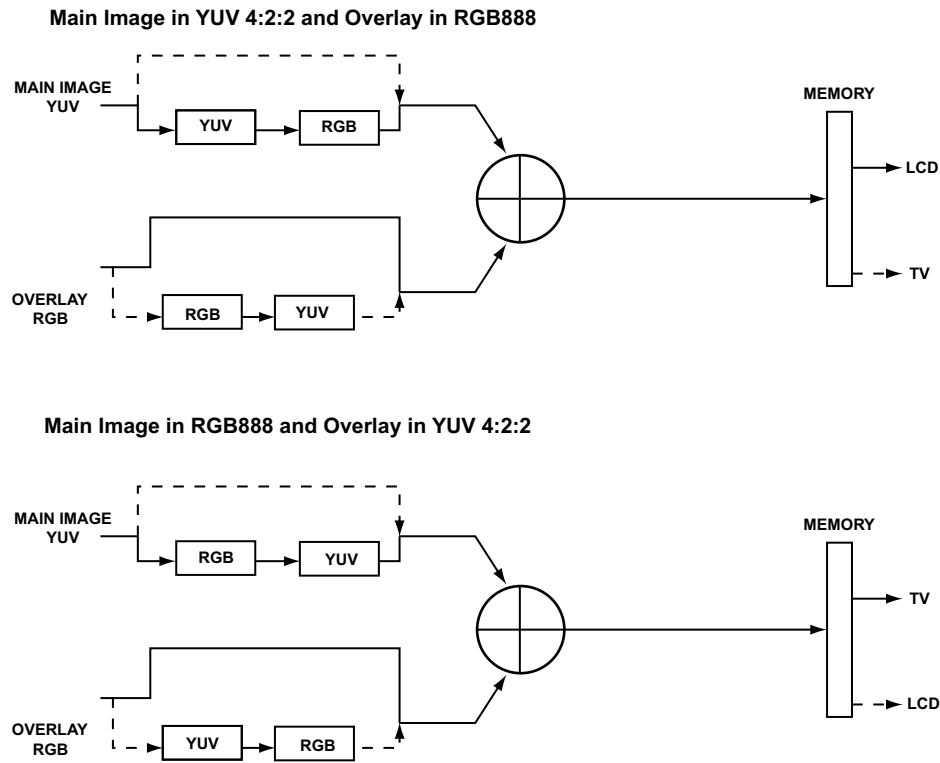


**Figure 32-11:** Image and Overlay in Different Formats

## PIXC Mode Case 3 - Color Space Conversion Only

In this case, there is no overlay blending. The main image is brought into the PIXC, the color space converted, and then sent back to memory. See figures below.

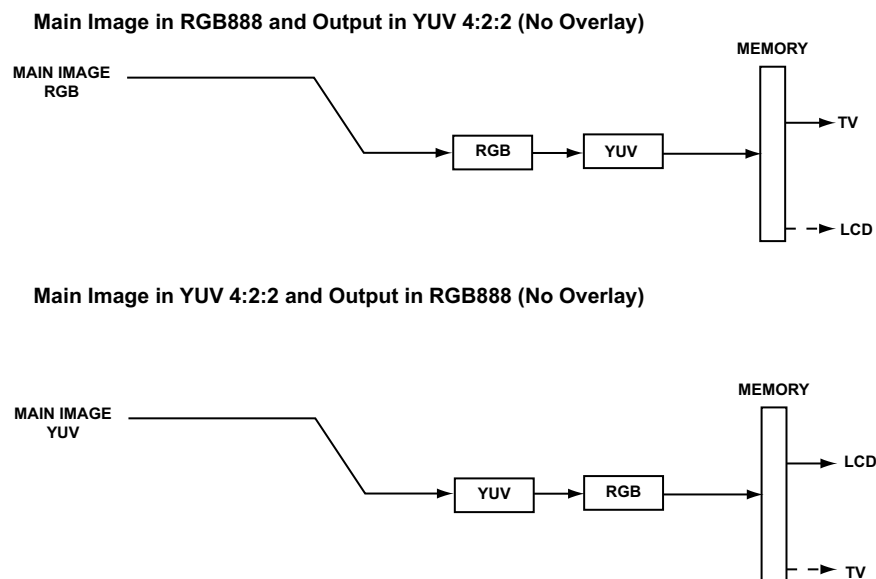


**Figure 32-12:** Color Space Conversion Only

For this mode, the register settings are: PIXC\_CTL.EN = 1, PIXC\_CTL.OVAEN = 0 and PIXC\_CTL.OVBEN = 0.

## PIXC Image/Overlay/Format Actions

The table below lists the PIXC actions that take place based on any possible combination of image, overlay, and output data formats. Note the following definitions used in the table.

- CSC = Color Space Conversion
- US = Up sampling
- DS = Down Sampling
- YUV = YUV 4:2:2 format
- RGB = RGB888 format

**Table 32-7:** PIXC Actions

| Image Data Format | Overlay Data Format | Output Data Format | PIXC Actions                                            |
|-------------------|---------------------|--------------------|---------------------------------------------------------|
| YUV               | No Overlay          | RGB                | US followed by CSC                                      |
| RGB               | No Overlay          | YUV                | CSC followed by DS                                      |
| YUV               | YUV                 | YUV                | US in both paths followed by DS before output           |
| YUV               | RGB                 | RGB                | US in image path, CSC in image path                     |
| YUV               | YUV                 | RGB                | US in both paths, followed by CSC                       |
| YUV               | RGB                 | YUV                | CSC in overlay path, US in image path, DS before output |
| RGB               | YUV                 | YUV                | CSC in image path, US in overlay path, DS before output |
| RGB               | YUV                 | RGB                | US in overlay path, CSC in overlay path                 |
| RGB               | RGB                 | YUV                | CSC followed by DS                                      |
| RGB               | RGB                 | RGB                | No CSC, No US, No DS                                    |



## PIXC Image/Overlay/Format Recommendations

For best results, the overlay should start on an odd-numbered pixel so that the U and V components of the image and the overlay are aligned. Otherwise artifacts may occur in the combined image.

When both the image and the overlay are in YUV 4:2:2 format and the output is also in YUV 4:2:2 format, the duplicating-dropping mode (`PIXC_CTL.UDSMOD`) is used to prevent a low-pass filtering effect on the images

## PIXC Image/Overlay/Format Special Use Cases

There are ways by which the PIXC can be made to operate on certain data formats that it does not support in any standard modes. For example, YUV 4:4:4 is similar to RGB 888 with respect to the number of pixels per 32-bit DMA word. So the PIXC can be configured to work with the YUV 4:4:4 data format by intelligently programming the `IFRMT`, `OVRMT`, and `OUTFRMT` bit fields in the `PIXC_CTL` register and the color space conversion coefficients.

These special usage cases are shown in the sections that follow.

### Example 1 - YUV 4:2:2 to YUV 4:4:4 or LCD/RGB

See the figures for special use of this mode:

- `IFRMT` = YUV
- `OVRMT` = YUV
- `OUTFRMT` = RGB

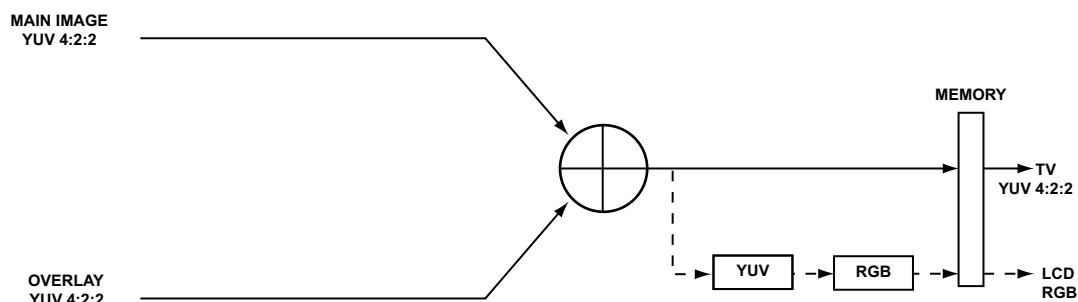


**Figure 32-13:** YUV 4:2:2/YUV 4:2:2 to YUV 4:2:2 or LCD/RGB

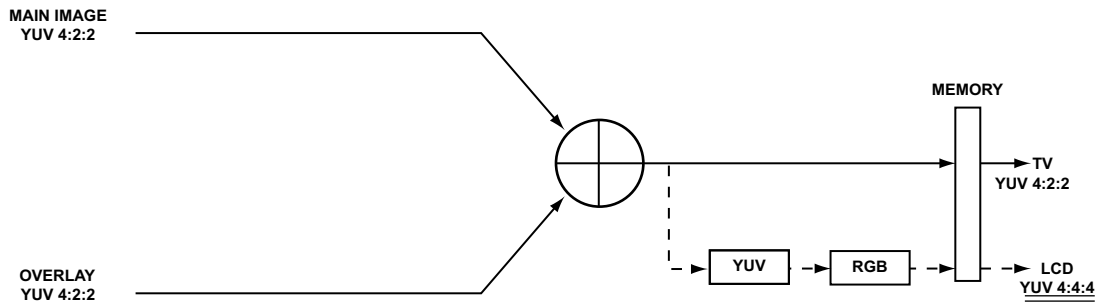


**Figure 32-14:** YUV 4:2:2/YUV 4:2:2 to YUV 4:2:2 or YUV 4:4:4

## Example 2 - YUV 4:4:4 to YUV 4:4:4 or YUV 4:2:2

See the figures for special use of this mode.

In the special usage of this mode, YUV 4:4:4 input produces a blended YUV 4:2:2 or YUV 4:4:4 data stream. A CSC matrix with coefficients of 1 is needed.

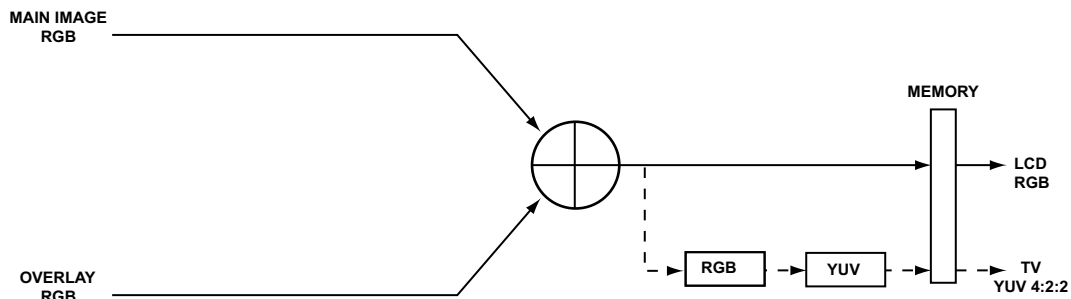


**Figure 32-15:** RGB/RGB to LCD/RGB or YUV 4:2:2

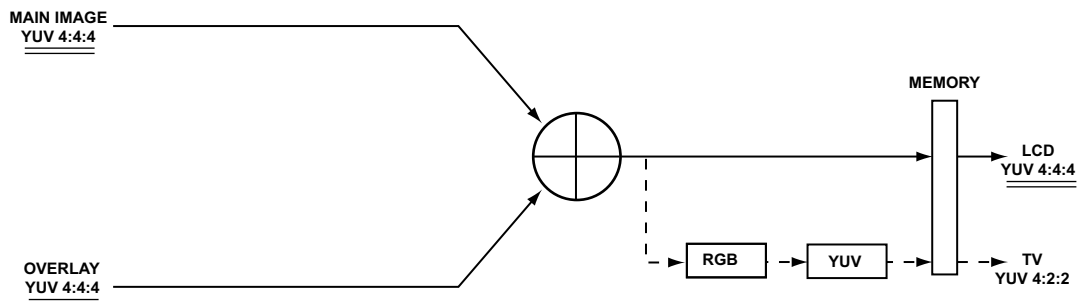


**Figure 32-16:** YUV 4:4:4/YUV 4:4:4 to YUV 4:4:4 or YUV 4:2:2

## Example 3 - YUV 4:2:2/4:4:4 to YUV 4:4:4 or YUV 4:2:2

See the figures for special use of this mode.

In the special usage of this mode, a YUV 4:4:4 input stream and a YUV 4:2:2 input stream can be blended to produce either a YUV 4:4:4 or a YUV 4:2:2 output stream.

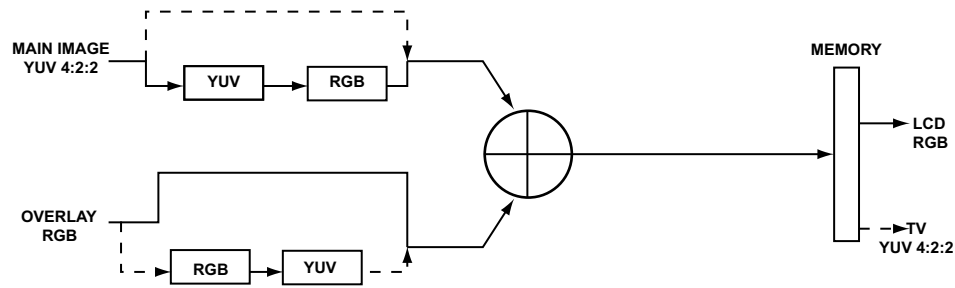


**Figure 32-17:** YUV 4:2:2/RGB to LCD/RGB or YUV 4:2:2

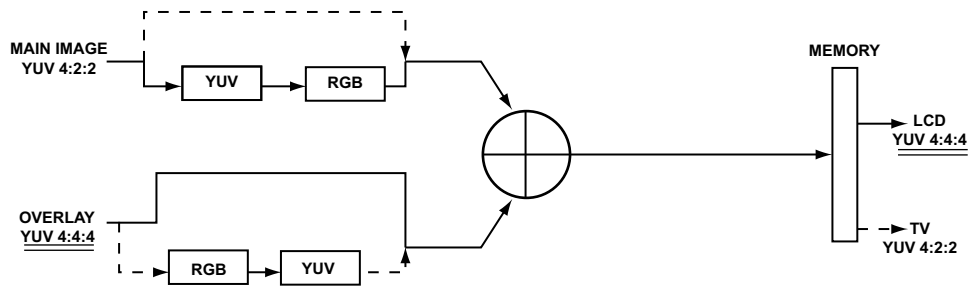


**Figure 32-18:** YUV 4:2:2 / YUV 4:4:4 to YUV 4:4:4 or YUV 4:2:2

### Example 4 - YUV 4:4:4 to YUV 4:2:2

See the figures for special use of this mode:

- PIXC\_EN = 1
- OVR\_A\_EN = 0
- OVR\_B\_EN = 0
- IMG\_FORM = RGB
- OUT\_FORM = YUV
- All CSC coefficients = 1

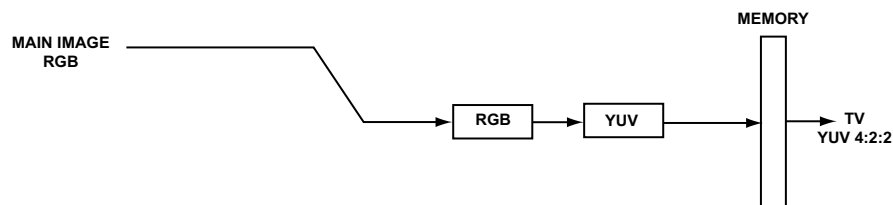


**Figure 32-19:** RGB to YUV 4:2:2

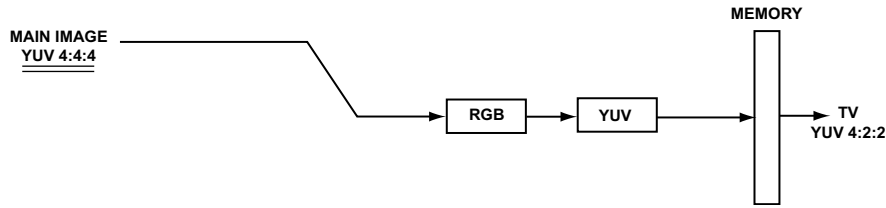


Figure 32-20: YUV 4:4:4 to YUV 4:2:2

## Color Space Conversion Matrix Equations

The PIXC color space conversion block implements the following matrix equation.

$$K \times \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} + \begin{bmatrix} A_{14} \\ A_{24} \\ A_{34} \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix}$$

The  $A_{xx}$  coefficients are 10-bit signed values represented in two's complement format.  $A_{11} \dots A_{33}$  are coefficient multipliers (for most cases, it is sufficient to specify these as integers between  $-512$  and  $511$ ), and  $A_{14}$ ,  $A_{24}$ ,  $A_{34}$  are simply offsets added to the result for each row.  $B_1$ ,  $B_2$ ,  $B_3$  represent the input pixel component values (for example, YUV or RGB) and  $C_1$ ,  $C_2$ ,  $C_3$  are the output pixel component values. Output pixel values are rounded to the nearest integer.

The constant  $K$  equals  $1/512$ . For example, to set  $A_{11}$ 's effective value to  $0.299$ , this coefficient's MMR should be programmed to  $\text{ROUND}(.299 \times 512)$ , or  $153$ . If a coefficient needs to be programmed with a value greater than  $1$ , an extra bit exists in each coefficient's MMR to specify if an extra multiply by  $4$  must be performed after multiplying the input value by its coefficient. However, this setting can only be specified for an entire row, so if this bit is set, all the coefficients for that row ( $A_{x1} - A_{x3}$ ) should be calculated as  $\text{ROUND}(\text{coeff} \times 512/4)$ . In other words, the constant  $K$  effectively becomes  $1/128$  for that row.

For reference, the matrix equations representing conversion between YUV and RGB formats are:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168 & -0.330 & 0.498 \\ 0.498 & -0.417 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.397 \\ 1.000 & -0.343 & -0.711 \\ 1.000 & 1.765 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} + \begin{bmatrix} -179 \\ 135 \\ -226 \end{bmatrix}$$

**NOTE:** For YUV-to-RGB conversion, the PIXC expects the input data to be arranged in the following order: VYUY, VYUY, and so on. As a result, if the input data is instead arranged as UYVY, UYVY, and so on, then, the columns  $A_{x2}$  and  $A_{x3}$  of the coefficient matrix are swapped.

For RGB-to-YUV conversion, the PIXC arranges the output data by default in the following order: VYUY, VYUY, and so on. If the output data is desired to instead be arranged as UYVY, UYVY, and so on, then, rows A2x and A3x of the coefficient and bias matrices are swapped.

## Color Space Converter Output Thresholds

Each PIXC output sample is 8 bits wide, whether it is an R, G, B, Y, U or V component value. Therefore, any output sample must be in the 0 to 255 range. Since all the coefficients are programmable, some of the inputs, when operated upon by the coefficients, may produce an output outside the 0 to 255 range. In such cases, the PIXC clips the output component's value to 0 or 255.

## YUV Re-Sampling

When the color space converter operates between two color spaces, it requires all components of each pixel to be present in the data stream.

Therefore, the PIXC internally up samples the YUV 4:2:2 data stream before a YUV-to-RGB conversion and similarly down samples the YUV 4:4:4 data stream after a RGB-to-YUV conversion. The re-sampling always takes place between YUV 4:2:2 and YUV 4:4:4 formats, but a certain flexibility is provided with regard to how the re-sampling is done by the PIXC in each case.

## Up Sampling

A YUV 4:2:2-to-YUV 4:4:4 conversion can be performed either by averaging or by duplicating the pixel components. The UDSDMOD bit in the PIXC\_CTL register specifies the up sampling mode. The default setting of this bit is 0, which corresponds to duplication of the chroma components (Us and Vs) from the odd pixels to the even pixels:

|                       |                                     |
|-----------------------|-------------------------------------|
| YUV 4:2:2 input:      | V1Y1, U1Y2, V3Y3, U3Y4, ...         |
| YUV 4:4:4 conversion: | Y1U1V1, Y2U1V1, Y3U3V3, Y4U3V3, ... |

Setting the PIXC\_CTL.UDSDMOD bit to 1 enables the averaging of the chroma components of the preceding and succeeding pixels to obtain the intermediate chroma value. In other words, two consecutive odd-numbered pixels' chroma components are averaged to obtain the intermediate even-numbered pixel's chroma components:

|                       |                                                                                                                   |
|-----------------------|-------------------------------------------------------------------------------------------------------------------|
| YUV 4:2:2 input:      | V1Y1, U1Y2, V3Y3, U3Y4, ...                                                                                       |
| YUV 4:4:4 conversion: | Y1U1V1, Y2U2V2 [ $U2=(U1+U3)/2$ ,<br>$V2=(V1+V3)/2$ ], Y3U3V3, Y4U4V4<br>[ $U4=(U3+U5)/2$ , $V4=(V3+V5)/2$ ], ... |

If the sum of the preceding and succeeding pixels' U/V components is an odd number, the average is rounded down (truncated to an integer value).

Since the last pixel on a line is always an even-numbered pixel, the last odd pixel value on that line is used as the last even pixel value during up sampling.

## Down Sampling

A YUV 4:4:4-to-YUV 4:2:2 conversion can be performed either by averaging or by dropping the pixel components. The `PIXC_CTL.UDSMOD` bit also governs the down sampling mode. Setting the `PIXC_CTL.UDSMOD` bit to 0 (default) enables the dropping of the chroma components of the even numbered pixels.

|                       |                                     |
|-----------------------|-------------------------------------|
| YUV 4:4:4 input:      | Y1U1V1, Y2U2V2, Y3U3V3, Y4U4V4, ... |
| YUV 4:2:2 conversion: | V1Y1, U1Y2, V3Y3, U3Y4, ...         |

Setting the `PIXC_CTL.UDSMOD` bit to 1 enables the averaging of the chroma components of two consecutive pixels to obtain a single chroma value for a pixel pair.

## PIXC Supported Data Formats

For the implementation of overlay, the PIXC needs two input data streams from two separate data buffers; a main image buffer and an overlay buffer. The data in the main image buffer can be formatted as YUV 4:2:2, RGB888, RGB666 or RGB565 while the overlay data must be in YUV 4:2:2 or RGB888 format. The main image data and the overlay data can be in different formats. The output data can also be formatted at YUV 4:2:2, RGB888, RGB666 or RGB565.

## Operation in YUV 4:2:2 Format

Each Y/U/V component is stored in 8 bits of data. The PIXC only accepts a YUV 4:2:2 interleaved format, in the following sequence:

V1, Y1, U1, Y2, V3, Y3, U3, Y4 ...

(Two components with the same suffix number (for example, V1 and U1) implies that they are extracted from the same pixel.)

It is the user's responsibility to ensure that the YUV source data to the PIXC is in the correct interleaved format. Therefore, data processing may be necessary in order to meet this requirement.

The figures below illustrate correct PIXC input buffer structure and data stream format.

YUV 4:2:2 Data in an Interleaved Data Buffer Structure

|                          |        |        |        |        |  |          |          |
|--------------------------|--------|--------|--------|--------|--|----------|----------|
| ADDR_BASE + 0            | V1 1   | Y1 1   | U1 1   | Y1 2   |  | U1 639   | Y1 640   |
| ADDR_BASE + 1280         | V2 1   | Y2 1   | U2 1   | Y2 2   |  | U2 639   | Y2 640   |
|                          |        |        |        |        |  |          |          |
|                          |        |        |        |        |  |          |          |
| ADDR_BASE + (479 x 1280) | V480 1 | Y480 1 | U480 1 | Y480 2 |  | U480 639 | Y480 640 |

YUV 4:2:2 Expected Data Stream Format to PIXC

| B31 | B16 | B15 | B0  |
|-----|-----|-----|-----|
| Y2  | U1  | Y1  | V1  |
| Y4  | U3  | Y3  | V3  |
| ... | ... | ... | ... |
| ... | ... | ... | ... |

**Figure 32-21:** YUV 4:2:2 Expected Buffer Structure and Data Stream Format

The number of pixels per line in YUV mode must be an even number (for both input buffers), and the first chroma component in each line must be a V component.

## Operation in RGB888 Format

Each R/G/B component is stored in 8 bits of data. The figures below illustrate correct PIXC input buffer structure and data stream format.

RGB888 Data Expected Data Buffer Structure

|                          |        |      |      |      |  |          |          |
|--------------------------|--------|------|------|------|--|----------|----------|
| ADDR_BASE + 0            | R1 1   | G1 1 | B1 1 | R1 2 |  | G1 640   | B1 640   |
| ADDR_BASE + 1280         | R2 1   | R2 1 | B2 1 | B2 2 |  | G2 640   | B2 640   |
|                          |        |      |      |      |  |          |          |
|                          |        |      |      |      |  |          |          |
| ADDR_BASE + (479 x 1280) | R480 1 |      |      |      |  | U480 639 | Y480 640 |

RGB888 Expected Data Stream Format to PIXC

| B31 | B16 | B15 | B0  |
|-----|-----|-----|-----|
| R12 | B11 | G11 | R11 |
| G13 | R13 | B12 | G12 |
| ... | ... | ... | ... |
| ... | ... | ... | ... |

**Figure 32-22:** RGB888 Expected Buffer Structure and Data Stream Format

For operation in RGB format, the total number of pixels in both input buffers must be a multiple of 4, so that the image boundary aligns with a 32-bit DMA word boundary.

## Operation in RGB565 Format

Two pixels are packed into one 32 bit word. Each pixel occupies 16 bits. The figures below illustrates how the pixel data is arranged in memory.

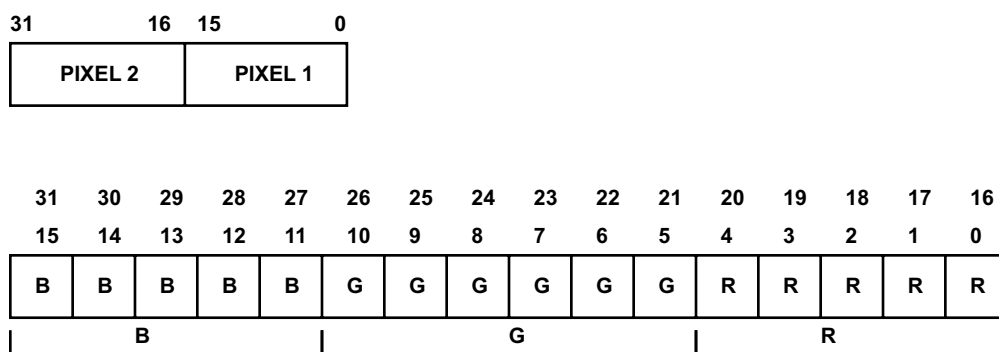


**Figure 32-23:** Memory Arrangement – RGB565 Format

## Operation in RGB666 Format

Four pixels are packed into three 32 bit words. Each pixel occupies 24 bits. Of these 24 bits, the lowest 18 bits are the pixel data and the highest 6 bits are discarded for inputs and zeroed for outputs. The figure



below illustrates how the pixel data is arranged in memory. Note that 1 = pixel 1, 2 = pixel 2, 3 = pixel 3, and 4 = pixel 4.

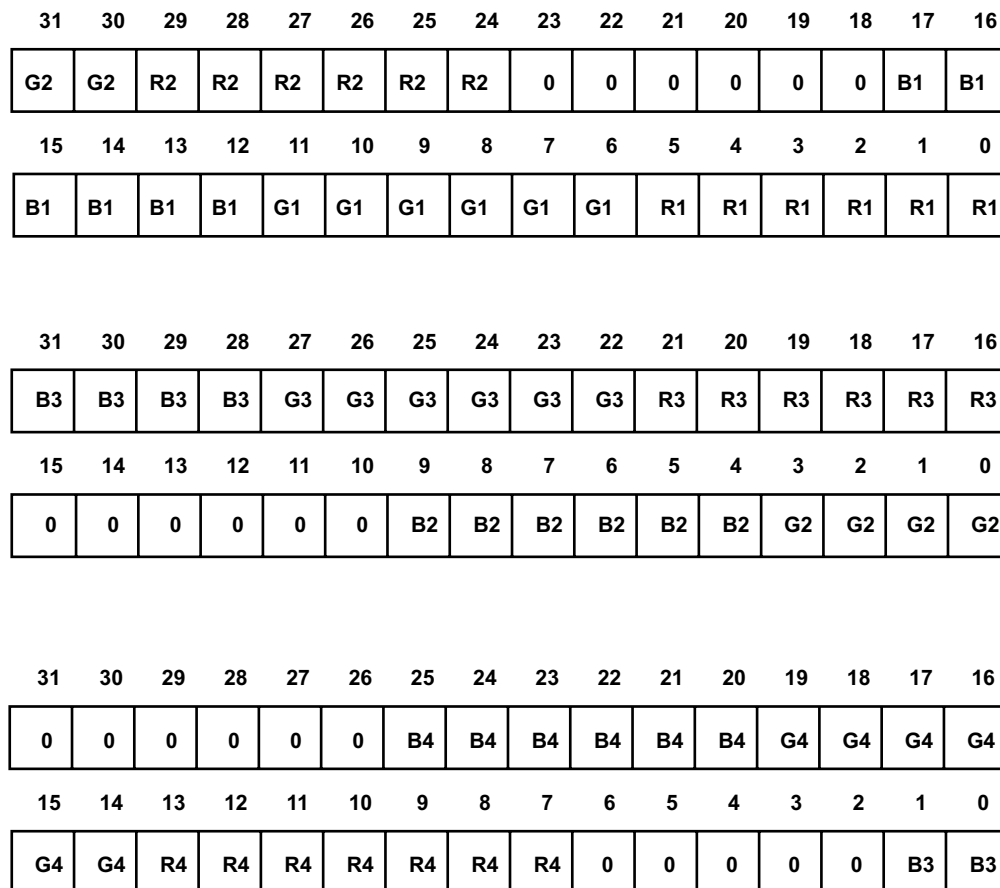


**Figure 32-24:** Memory Arrangement – RGB666 Format

## Operation with RGB656 and RGB666 Formats

For RGB656 and RGB666 formats, the PIXC internal engine reads the relevant bits from the input and zero pads to 8 bits. The internal PIXC data path is always 8 bits. After the data path operations (mixer, color space conversion), if the output requires the pixels in either RGB565 or RGB666, the individual color components are truncated (the lower bits dropped) and converted to the required formats.

## PIXC Interrupt Signals

There are three DDE's associated with the PIXC. There is one for the input data, one for the input overlay frame and one for the output frame. The interrupts generated are due to the configuration of the DDE. For instance, the DMA channel that puts the results back to memory can be configured to generate an interrupt after every frame is completed. For more information, refer to the DDE section of the manual.

**Table 32-8:** Interrupt List for the PIXC Module

| Trigger | ID  | Module | Count | Type  | Description   | DMA Channel |
|---------|-----|--------|-------|-------|---------------|-------------|
| STAT    | 110 | PIXC   | 0     | LEVEL | Status        |             |
| CH1_DMA | 108 | PIXC   | 0     | LEVEL | Channel 1 DMA | 36          |
| CH0_DMA | 107 | PIXC   | 0     | LEVEL | Channel 0 DMA | 35          |
| CH2_DMA | 109 | PIXC   | 0     | LEVEL | Channel 2 DMA | 37          |

## Trigger List for PIXC module

**Table 32-9:** Trigger List for the PIXC Module

| Interrupt | ID | Module | Count | Type       | Description   | MS Type |
|-----------|----|--------|-------|------------|---------------|---------|
| CH1_DMA   | 58 | PIXC   | 0     | PULSE/EDGE | Channel 1 DMA | MASTER  |
| CH2_DMA   | 59 | PIXC   | 0     | PULSE/EDGE | Channel 2 DMA | MASTER  |
| CH0_DMA   | 57 | PIXC   | 0     | PULSE/EDGE | Channel 0 DMA | MASTER  |
| CH1_DMA   | 58 | PIXC   | 0     | N/A        | Channel 1 DMA | SLAVE   |
| CH2_DMA   | 59 | PIXC   | 0     | N/A        | Channel 2 DMA | SLAVE   |
| CH0_DMA   | 57 | PIXC   | 0     | N/A        | Channel 0 DMA | SLAVE   |

## PIXC Programming Model

The PIXC is able to take data in from either memory or the video subsystem. The output is able to go back to memory or piped back to the video subsystem. Thus, video subsystem needs to be configured correctly before enabling the use of the PIXC.

## PIXC Mode Configuration

This section describes typical mode configuration tasks for using the PIXC.

### Performing Data Overlay

Data overlay allows a separate image to be overlaid or blended with the main input image.

1. Define rectangular region in the input image to be covered by an overlay.
2. Define single DMA descriptor for the overlay transfer.
3. Configure overlay coordinate registers in PIXC (PIXC\_HSTART\_A, PIXC\_HEND\_A, PIXC\_VSTART\_A, and PIXC\_VEND\_A).
4. Configure transparency ratio register (PIXC TRANSP\_A).
5. Optionally define a second overlay region and configure PIXC\_HSTART\_B, PIXC\_HEND\_B, PIXC\_VSTART\_B, and PIXC\_VEND\_B and PIXC\_TRANS\_B accordingly.
6. Enable PIXC and overlay functionality by setting PIXC\_CTL.EN = 1, PIXC\_CTL.OVAEN = 1 and PIXC\_CTL.OVBEN = 1 (if a 2nd overlay region is defined).

*RESULT:*

The output image is blended with the overlay images at the specified overlay region.

### Performing Color Space Conversion Only

The PIXC can be used to color convert an image or frame from RGB to YUV or from YUV to RGB

1. Configure and set up input connection of the PIXC in the video sub-system to obtain the main input image.
2. Configure and set up output connection of the PIXC in the video sub-system to output the resulting output image.
3. Program the conversion component registers (PIXC\_CONRY, PIXC\_CONGU and PIXC\_CONBV) and conversion bias register (PIXC\_CCBIAS).
4. Enable the PIXC by setting PIXC\_CTL.EN = 1 but keeping the overlay enables off by setting PIXC\_CTL.OVAEN = 0 and PIXC\_CTL.OVBEN = 0.

*RESULT:*

Converts an input image from RGB to YUV or an image from YUV to RGB

## PIXC Register Descriptions

Pixel Compositor (PIXC) contains the following registers.

**Table 32-10:** PIXC Module Registers

| Name          | Description                           |
|---------------|---------------------------------------|
| PIXC_CTL      | Control Register                      |
| PIXC_PPL      | Pixels Per Line Register              |
| PIXC_LPF      | Line Per Frame Register               |
| PIXC_HSTART_A | Overlay A Horizontal Start Register   |
| PIXC_HEND_A   | Overlay A Horizontal End Register     |
| PIXC_VSTART_A | Overlay A Vertical Start Register     |
| PIXC_VEND_A   | Overlay A Vertical End Register       |
| PIXC_TRANSP_A | Overlay A Transparency Ratio Register |
| PIXC_HSTART_B | Overlay B Horizontal Start Register   |
| PIXC_HEND_B   | Overlay B Horizontal End Register     |
| PIXC_VSTART_B | Overlay B Vertical Start Register     |
| PIXC_VEND_B   | Overlay B Vertical End Register       |
| PIXC_TRANSP_B | Overlay B Transparency Ratio Register |
| PIXC_IRQSTAT  | Interrupt Status Register             |
| PIXC_CONRY    | RY Conversion Component Register      |
| PIXC_CONGU    | GU Conversion Component Register      |
| PIXC_CONBV    | BV Conversion Component Register      |
| PIXC_CCBIAS   | Conversion Bias Register              |
| PIXC_TC       | Transparency Color Register           |
| PIXC_REVID    | Revision Id                           |

## Control Register

The PIXC\_CTL register provides module enable, overlay enable, re-sampling mode selection, input/overlay/output data format selection, transparent color enable, and input/output RGB data format.

### PIXC\_CTL: Control Register - R/W

Reset = 0x0000 0000

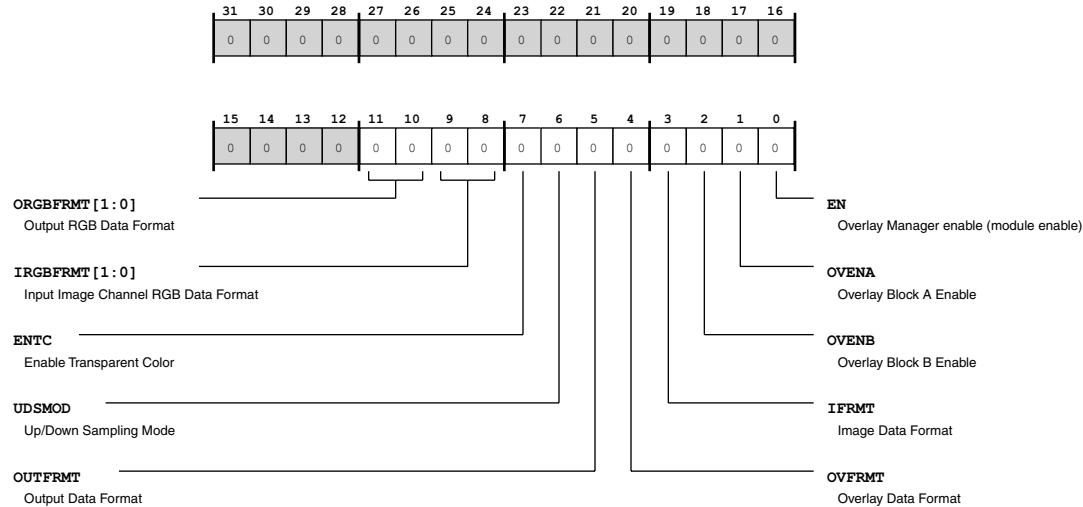


**Figure 32-25:** PIXC\_CTL Register Diagram

**Table 32-11:** PIXC\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 11:10   | ORGBFRMT | Output RGB Data Format.<br>The PIXC_CTL.ORGBFRMT bits select the output RGB data format. This selection only is relevant when PIXC_CTL.OUTFRMT is set. |
|         |          | 0 RGB 888                                                                                                                                              |
|         |          | 1 Reserved                                                                                                                                             |
|         |          | 2 RGB 565                                                                                                                                              |
|         |          | 3 RGB 666                                                                                                                                              |

**Table 32-11:** PIXC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 9:8     | IRGBFRMT | Input Image Channel RGB Data Format.<br>The <code>PIXC_CTL.IRGBFRMT</code> bits select the input image channel RGB data format. This selection only is relevant when <code>PIXC_CTL.IFRMT</code> is set. |
|         |          | 0      RGB 888                                                                                                                                                                                           |
|         |          | 1      Reserved                                                                                                                                                                                          |
|         |          | 2      RGB 565                                                                                                                                                                                           |
|         |          | 3      RGB 666                                                                                                                                                                                           |
| 7       | ENTC     | Enable Transparent Color.                                                                                                                                                                                |
|         |          | 0      Disable Transparent Color                                                                                                                                                                         |
|         |          | 1      Enable Transparent Color                                                                                                                                                                          |
| 6       | UDSMOD   | Up/Down Sampling Mode.<br>The <code>PIXC_CTL.UDSMOD</code> selects whether the PIXC either uses averaging for up/down sampling or uses duplicating for up sampling and dropping for down sampling.       |
|         |          | 0      Duplicating for up sampling and dropping for down sampling                                                                                                                                        |
|         |          | 1      Averaging for both up sampling and down sampling                                                                                                                                                  |
| 5       | OUTFRMT  | Output Data Format.<br>The <code>PIXC_CTL.OUTFRMT</code> selects YUV or RGB for the output format.                                                                                                       |
|         |          | 0      YUV Data Format                                                                                                                                                                                   |
|         |          | 1      RGB Data Format                                                                                                                                                                                   |
| 4       | OVFRMT   | Overlay Data Format.<br>The <code>PIXC_CTL.OVFRMT</code> selects YUV or RGB for the overlay format.                                                                                                      |
|         |          | 0      YUV Data Format                                                                                                                                                                                   |
|         |          | 1      RGB Data Format                                                                                                                                                                                   |

**Table 32-11:** PIXC\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                             |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | IFRMT    | Image Data Format.<br>The <code>PIXC_CTL.IFRMT</code> selects YUV or RGB for the input format.                                                      |
|         |          | 0 YUV Data Format                                                                                                                                   |
|         |          | 1 RGB Data Format                                                                                                                                   |
| 2       | OVENB    | Overlay Block B Enable.<br>If both <code>PIXC_CTL.OVENA</code> and <code>PIXC_CTL.OVENB</code> are cleared, only color space conversion is enabled. |
|         |          | 0 Disable                                                                                                                                           |
|         |          | 1 Enable                                                                                                                                            |
| 1       | OVENA    | Overlay Block A Enable.<br>If both <code>PIXC_CTL.OVENA</code> and <code>PIXC_CTL.OVENB</code> are cleared, only color space conversion is enabled. |
|         |          | 0 Disable                                                                                                                                           |
|         |          | 1 Enable                                                                                                                                            |
| 0       | EN       | Overlay Manager enable (module enable).<br>The <code>PIXC_CTL.EN</code> bit enables the PIXC.                                                       |
|         |          | 0 Disable                                                                                                                                           |
|         |          | 1 Enable                                                                                                                                            |

## Pixels Per Line Register

The `PIXC_PPL` register provides the number of pixels per line of the display.

**PIXC\_PPL: Pixels Per Line Register - R/W**

Reset = 0x0000

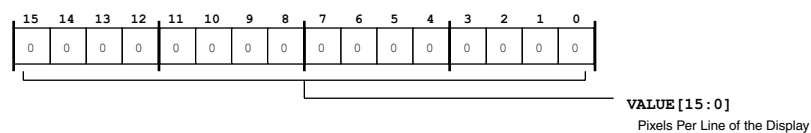


**Figure 32-26:** PIXC\_PPL Register Diagram

Table 32-12: PIXC\_PPL Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 15:0    | VALUE    | Pixels Per Line of the Display. |

Line Per Frame Register

The PIXC\_LPF register provides the number of lines per frame of the display.

PIXC\_LPF: Line Per Frame Register - R/W

Reset = 0x0000

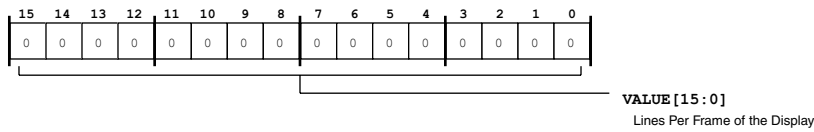


Figure 32-27: PIXC\_LPF Register Diagram

Table 32-13: PIXC\_LPF Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 15:0    | VALUE    | Lines Per Frame of the Display. |

Overlay A Horizontal Start Register

The PIXC\_HSTART\_A register provides the horizontal start pixel coordinates of overlay A data.

PIXC\_HSTART\_A: Overlay A Horizontal Start Register - R/W

Reset = 0x0000

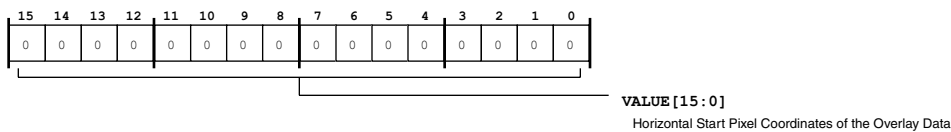


Figure 32-28: PIXC\_HSTART\_A Register Diagram



**Table 32-14:** PIXC\_HSTART\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                                 |
|---------|----------|---------------------------------------------------------|
| 15:0    | VALUE    | Horizontal Start Pixel Coordinates of the Overlay Data. |

## Overlay A Horizontal End Register

The PIXC\_HEND\_A registers provides the horizontal end pixel coordinates of overlay A data.

**PIXC\_HEND\_A: Overlay A Horizontal End Register - R/W**

Reset = 0x0000

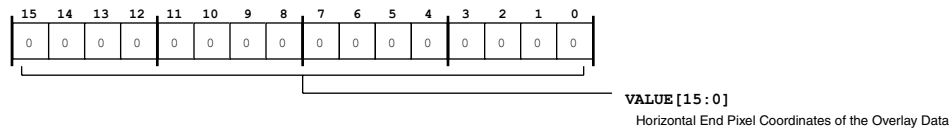


**Figure 32-29:** PIXC\_HEND\_A Register Diagram

**Table 32-15:** PIXC\_HEND\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 15:0    | VALUE    | Horizontal End Pixel Coordinates of the Overlay Data. |

## Overlay A Vertical Start Register

The PIXC\_VSTART\_A registers provides the vertical start pixel coordinates of overlay A data.

**PIXC\_VSTART\_A: Overlay A Vertical Start Register - R/W**

Reset = 0x0000

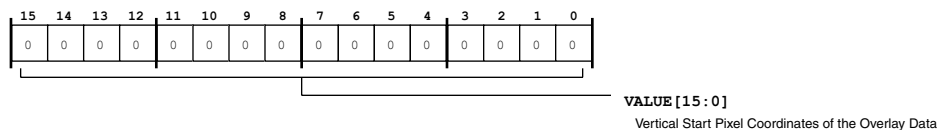


**Figure 32-30:** PIXC\_VSTART\_A Register Diagram

Table 32-16: PIXC\_VSTART\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 15:0    | VALUE    | Vertical Start Pixel Coordinates of the Overlay Data. |

Overlay A Vertical End Register

The PIXC\_VEND\_A register provides the vertical end pixel coordinates of overlay A data.

PIXC\_VEND\_A: Overlay A Vertical End Register - R/W

Reset = 0x0000

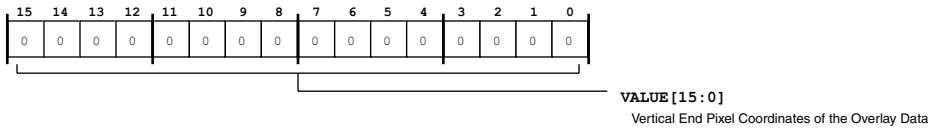


Figure 32-31: PIXC\_VEND\_A Register Diagram

Table 32-17: PIXC\_VEND\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 15:0    | VALUE    | Vertical End Pixel Coordinates of the Overlay Data. |

Overlay A Transparency Ratio Register

The PIXC TRANSP\_A register provides overlay A transparency ratio values.

PIXC\_TRANSP\_A: Overlay A Transparency Ratio Register - R/W

Reset = 0x000f

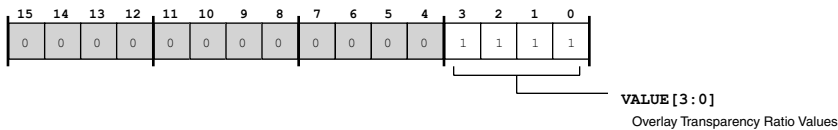


Figure 32-32: PIXC\_TRANSP\_A Register Diagram

**Table 32-18:** PIXC TRANSP\_A Register Fields

| Bit No. | Bit Name | Description/Enumeration            |
|---------|----------|------------------------------------|
| 3:0     | VALUE    | Overlay Transparency Ratio Values. |

## Overlay B Horizontal Start Register

The PIXC\_HSTART\_B register provides the horizontal start pixel coordinates of overlay B data.

**PIXC\_HSTART\_B: Overlay B Horizontal Start Register - R/W**

Reset = 0x0000

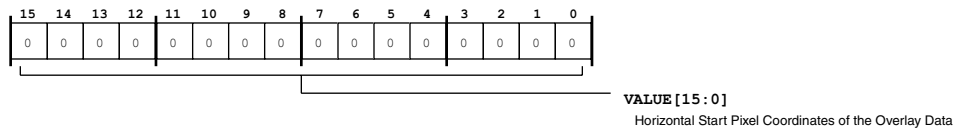


**Figure 32-33:** PIXC\_HSTART\_B Register Diagram

**Table 32-19:** PIXC\_HSTART\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                                 |
|---------|----------|---------------------------------------------------------|
| 15:0    | VALUE    | Horizontal Start Pixel Coordinates of the Overlay Data. |

## Overlay B Horizontal End Register

The PIXC\_HEND\_B registers provides the horizontal end pixel coordinates of overlay B data.

**PIXC\_HEND\_B: Overlay B Horizontal End Register - R/W**

Reset = 0x0000

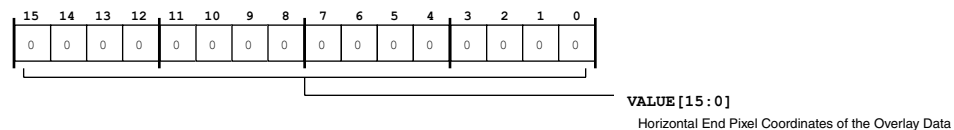


**Figure 32-34:** PIXC\_HEND\_B Register Diagram

Table 32-20: PIXC\_HEND\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 15:0    | VALUE    | Horizontal End Pixel Coordinates of the Overlay Data. |

Overlay B Vertical Start Register

The PIXC\_VSTART\_B registers provides the vertical start pixel coordinates of overlay B data.

PIXC\_VSTART\_B: Overlay B Vertical Start Register - R/W

Reset = 0x0000

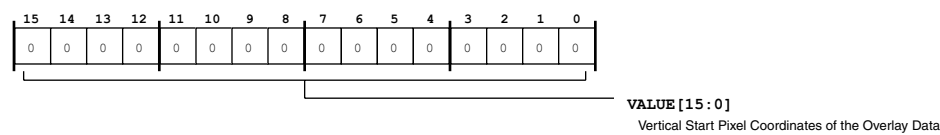


Figure 32-35: PIXC\_VSTART\_B Register Diagram

Table 32-21: PIXC\_VSTART\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                               |
|---------|----------|-------------------------------------------------------|
| 15:0    | VALUE    | Vertical Start Pixel Coordinates of the Overlay Data. |

Overlay B Vertical End Register

The PIXC\_VEND\_B register provides the vertical end pixel coordinates of overlay B data.

PIXC\_VEND\_B: Overlay B Vertical End Register - R/W

Reset = 0x0000

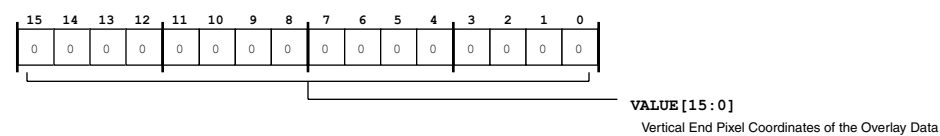


Figure 32-36: PIXC\_VEND\_B Register Diagram

**Table 32-22:** PIXC\_VEND\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration                             |
|---------|----------|-----------------------------------------------------|
| 15:0    | VALUE    | Vertical End Pixel Coordinates of the Overlay Data. |

## Overlay B Transparency Ratio Register

The PIXC TRANSP\_B register provides overlay B transparency ratio values.

**PIXC\_TRANSP\_B: Overlay B Transparency Ratio Register - R/W**

Reset = 0x000f

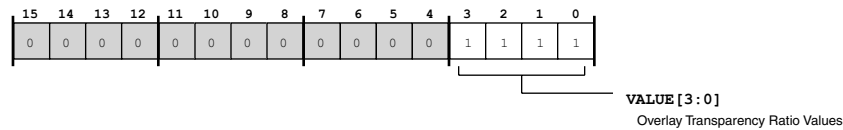


**Figure 32-37:** PIXC\_TRANSP\_B Register Diagram

**Table 32-23:** PIXC\_TRANSP\_B Register Fields

| Bit No. | Bit Name | Description/Enumeration            |
|---------|----------|------------------------------------|
| 3:0     | VALUE    | Overlay Transparency Ratio Values. |

## Interrupt Status Register

The PIXC\_IRQSTAT register provides overlay interrupt configuration and status information.

**PIXC\_IRQSTAT: Interrupt Status Register - R/W**

Reset = 0x0000

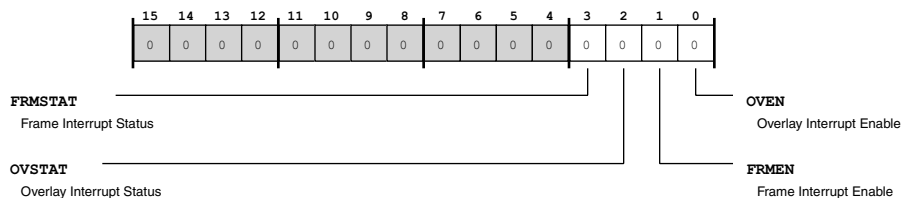


**Figure 32-38:** PIXC\_IRQSTAT Register Diagram

Table 32-24: PIXC\_IRQSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                  |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | FRMSTAT  | Frame Interrupt Status.<br>The PIXC_IRQSTAT.FRMSTAT indicates an end of frame condition for the PIXC. This status bit is write-1-to-clear.               |
| 2       | OVSTAT   | Overlay Interrupt Status.<br>The PIXC_IRQSTAT.OVSTAT indicates an end of last valid overlay condition for the PIXC. This status bit is write-1-to-clear. |
| 1       | FRMEN    | Frame Interrupt Enable.<br>The PIXC_IRQSTAT.FRMEN enables the PIXC end of frame interrupt.                                                               |
|         |          | 0                      Disable                                                                                                                           |
|         |          | 1                      Enable                                                                                                                            |
| 0       | OVEN     | Overlay Interrupt Enable.<br>The PIXC_IRQSTAT.OVEN enables the PIXC end of last valid overlay interrupt.                                                 |
|         |          | 0                      Disable                                                                                                                           |
|         |          | 1                      Enable                                                                                                                            |

RY Conversion Component Register

The PIXC\_CONRY register provides the R/Y conversion coefficients in the color space conversion matrix.

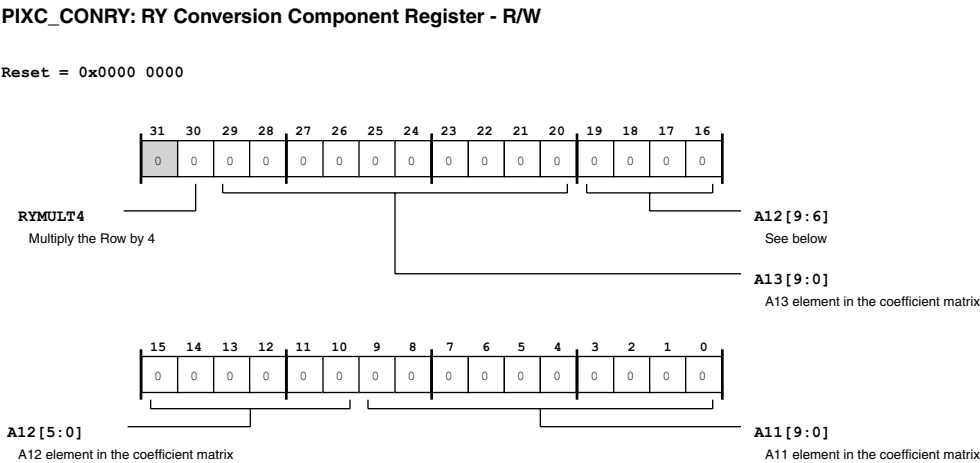


Figure 32-39: PIXC\_CONRY Register Diagram

**Table 32-25:** PIXC\_CONRY Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 30      | RYMULT4  | Multiply the Row by 4.                 |
|         |          | 0      Disable                         |
|         |          | 1      Enable                          |
| 29:20   | A13      | A13 element in the coefficient matrix. |
| 19:10   | A12      | A12 element in the coefficient matrix. |
| 9:0     | A11      | A11 element in the coefficient matrix. |

## GU Conversion Component Register

The PIXC\_CONGU register provides the G/U conversion coefficients in the color space conversion matrix.

### PIXC\_CONGU: GU Conversion Component Register - R/W

Reset = 0x0000 0000

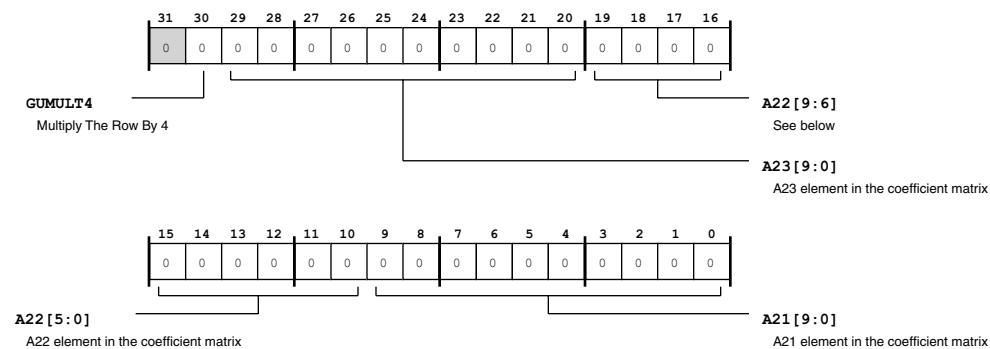


**Figure 32-40:** PIXC\_CONGU Register Diagram

**Table 32-26:** PIXC\_CONGU Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 30      | GUMULT4  | Multiply The Row By 4.  |
|         |          | 0      Disable          |
|         |          | 1      Enable           |

Table 32-26: PIXC\_CONGU Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                          |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:20   | A23      | A23 element in the coefficient matrix.                                                                                                           |
| 19:10   | A22      | A22 element in the coefficient matrix.                                                                                                           |
| 9:0     | A21      | A21 element in the coefficient matrix.<br>The PIXC_CONGU register provides the B/V conversion coefficients in the color space conversion matrix. |

BV Conversion Component Register

PIXC\_CONBV: BV Conversion Component Register - R/W

Reset = 0x0000 0000

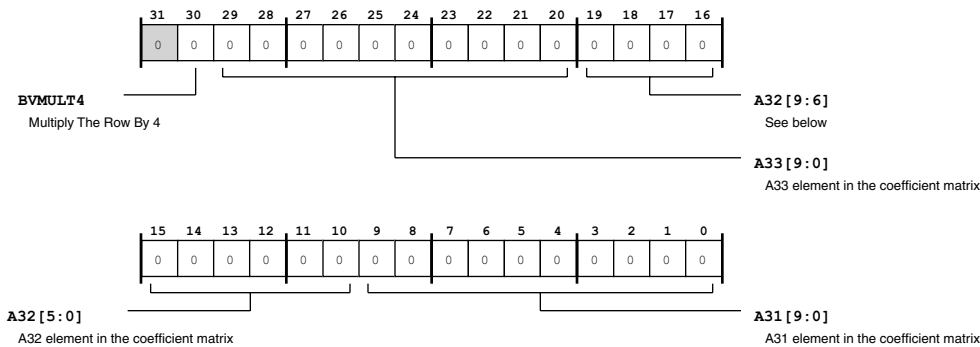


Figure 32-41: PIXC\_CONBV Register Diagram

Table 32-27: PIXC\_CONBV Register Fields

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 30      | BVMULT4  | Multiply The Row By 4.                 |
|         |          | 0 Disable                              |
|         |          | 1 Enable                               |
| 29:20   | A33      | A33 element in the coefficient matrix. |
| 19:10   | A32      | A32 element in the coefficient matrix. |



**Table 32-27:** PIXC\_CONBV Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                |
|---------|----------|----------------------------------------|
| 9:0     | A31      | A31 element in the coefficient matrix. |

Conversion Bias Register

The PIXC\_CCBIAS register provides the bias values in the color space conversion matrix.

**PIXC\_CCBIAS: Conversion Bias Register - R/W**

Reset = 0x0000 0000

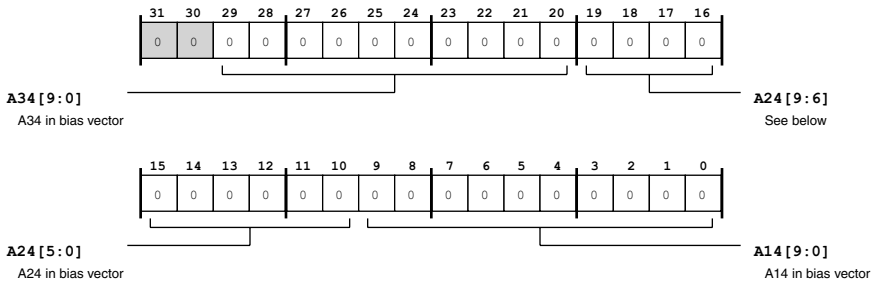


**Figure 32-42:** PIXC\_CCBIAS Register Diagram

**Table 32-28:** PIXC\_CCBIAS Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 29:20   | A34      | A34 in bias vector.     |
| 19:10   | A24      | A24 in bias vector.     |
| 9:0     | A14      | A14 in bias vector.     |

Transparency Color Register

The PIXC\_TC register provides the transparent color value.

PIXC\_TC: Transparency Color Register - R/W

Reset = 0x0000 0000

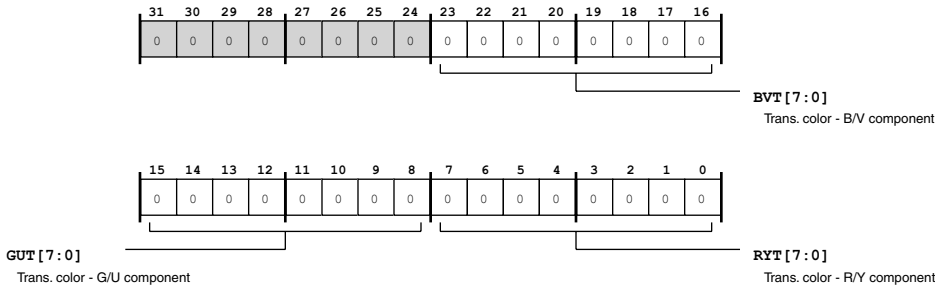


Figure 32-43: PIXC\_TC Register Diagram

Table 32-29: PIXC\_TC Register Fields

| Bit No. | Bit Name | Description/Enumeration       |
|---------|----------|-------------------------------|
| 23:16   | BVT      | Trans. color - B/V component. |
| 15:8    | GUT      | Trans. color - G/U component. |
| 7:0     | RYT      | Trans. color - R/Y component. |

Revision Id

The PIXC\_REVID register provides the PIXC module revision information.

PIXC\_REVID: Revision Id - R/NW

Reset = 0x0000 0020

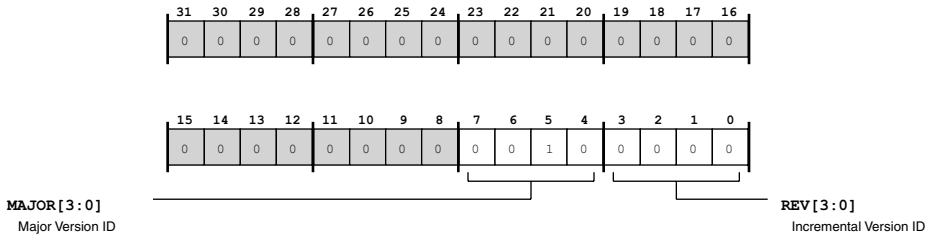


Figure 32-44: PIXC\_REVID Register Diagram

**Table 32-30:** PIXC\_REVID Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 7:4     | MAJOR    | Major Version ID.       |
| 3:0     | REV      | Incremental Version ID. |



## 33 Reset Control Unit (RCU)

Reset is the initial state of the whole processor or one of the cores and is the result of a hardware or software triggered event. In this state, all control registers are set to their default values and functional units are idle. Exiting a full system reset starts with Core-0 only being ready to boot. Exiting a Core n only reset starts with this Core n being ready to boot.

The reset control unit (RCU) controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints define how reset signals are generated. Programs must guarantee that none of the reset functions puts the system into an undefined state or causes resources to stall. This is particularly important when only one of the cores is reset (programs must ensure that there is no pending system activity involving the core that is being reset).

### RCU Register List

The reset control unit (RCU) controls how all the functional units in the processor enter and exit Reset. Differences in functional requirements and clocking constraints (units in different clock domains have to enter reset asynchronously, but units exit reset in a deterministic way) define how reset signals are generated. Reset signals propagate through all the functional units asynchronously. For more information on RCU functionality, see the RCU register descriptions.

**Table 33-1:** RCU Module Registers

| Name          | Description                       |
|---------------|-----------------------------------|
| RCU_CTL       | Control Register                  |
| RCU_STAT      | Status Register                   |
| RCU_CRCTL     | Core Reset Control Register       |
| RCU_CRSTAT    | Core Reset Status Register        |
| RCU_SIDIS     | System Interface Disable Register |
| RCU_SISTAT    | System Interface Status Register  |
| RCU_SVECT_LCK | SVECT Lock Register               |
| RCU_BCODE     | Boot Code Register                |

**Table 33-1:** RCU Module Registers (Continued)

| Name       | Description                |
|------------|----------------------------|
| RCU_SVECTn | Software Vector Register n |

## ADSP-BF60x RCU Trigger List

**Table 33-2:** Masters ADSP-BF60x RCU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
|             |    |             |

**Table 33-3:** Slaves ADSP-BF60x RCU Trigger List

| Description         | ID | Sensitivity |
|---------------------|----|-------------|
| RCU0 System Reset 0 | 0  |             |
| RCU0 System Reset 1 | 1  |             |
|                     |    |             |

## RCU Definitions

To make the best use of the RCU, it is useful to understand the terms in this section.

The following are Resets that are defined by their target

### Hardware Reset

All functional units are set to their default states, no exception. History is lost.

### System Reset

All functional units except the RCU are set to their default states.

### Core n Only Reset

Affects Core n only. The system software should guarantee that the core in reset state is not accessed by any bus master.

The following are Resets that are defined by their source

## Hardware Reset

The  $\overline{\text{RESET}}$  input signal is asserted active (pulled low).

## System Reset

May be triggered by software (writing to `RCU_CONTROL`) or by another functional unit such as the DPM (Hibernate) or any of the generic reset inputs.

## Core n Only Reset

Triggered by software.

# RCU Architectural Concepts

The following sections provide details about the Reset Control Unit Architecture.

## Reset Sources

The RCU provides the hardware that controls how all the functional units enter and exit reset. Differences in functional requirements and clocking constraints (for example units in different clock domains have to enter reset asynchronously but exit reset in a deterministic way) define how reset signals are generated. It is the program's responsibility to guarantee that none of the reset functions put the system in an undefined state or cause resources to stall. This is particularly important when only one of the cores is reset because the program needs to guarantee that there is no pending system activity involving Core n (for example no pending transactions to core n) before it is reset. The following table defines how reset sources affect the different functional units.

| Reset Source                           | Reset Type     | Affected Functional Units                                                                                                                                                       |
|----------------------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\overline{\text{RESET}}$              | Hardware Reset | All functional units except RTC (if present)                                                                                                                                    |
| Hibernate Reset                        | System Reset   | All functional units except RTC (if present), <code>RCU_STAT</code> and the units on the <code>VDDEXT</code> power domain                                                       |
| <code>SYSCLK</code> Clock Domain Reset | System Reset   | All functional units except RTC (if present), <code>RCU_STAT</code> , <code>RCU_BCODE</code> , <code>RCU_CRCTL.CRn</code> and the units on the <code>VDDEXT</code> power domain |
| <code>SCLK0</code> Clock Domain Reset  | System Reset   | All functional units except <code>RCU_STAT</code> , <code>RCU_BCODE</code> , <code>RCU_CRCTL.CRn</code> and the units on the <code>VDDEXT</code> power domain                   |

| Reset Source                                                               | Reset Type      | Affected Functional Units                                                                                                 |
|----------------------------------------------------------------------------|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| SCLK1 Clock Domain Reset                                                   | System Reset    | All functional units except RTC (if present), RCU_STAT, RCU_BCODE, RCU_CRCTL.CRn and the units on the VDDEXT power domain |
| RCU_CTL.SYSRST<br>System Reset (Software Triggered Reset)                  | System Reset    | All functional units except RTC (if present), RCU_STAT, RCU_BCODE, RCU_CRCTL.CRn and the units on the VDDEXT power domain |
| RCU_CRCTL.CRn bit set<br>(Software Triggered Reset) ( $15 \geq n \geq 0$ ) | Core Only Reset | Core n Only                                                                                                               |

## RCU Status and Error Signals

The RCU\_STAT register reflects status and error information. There are three kinds of errors that can occur in the RCU. The Reset Out error is triggered if RSTOUT is both asserted and deasserted at the same time. The Lock Write error occurs if an attempt is made to write a lock RCU register. The Address Error occurs if a read only register is written to or if an attempt is made to a reserved address within the RCU MMR address range.

## Resetting a Core

The RCU allows reset of a given core n using another core.

### PREREQUISITE:

Core n can be individually reset by software, either setting any of CRn ( $15 \geq n \geq 0$ ) bits in RCU\_CRCTL register. Cores that reset themselves will not be able to guarantee that all the system transactions to or from it have completed. Although a core n reset can be triggered by core n itself it is recommended that it is triggered by another core. Core n may be reset to restore its functionality when it cannot execute software. The suggested sequence to reset core n only is shown below.

1. Clear the RCU\_CRSTAT.CRn bit.
2. Disable interrupts to core n
3. Set the RCU\_SIDIS.SIn bit to disable core n's interfaces in order to stop DMA accesses to its L1, stop core n's accesses to memory, and stop accesses to MMRs.
4. Test the RCU\_SISTAT.SIn bit to detect when accesses to core n have been disabled and all the pending transactions have completed.
5. Set the RCU\_CRCTL.Rn bit to reset core n.



6. Poll the `RCU_CRSTAT.CRn` bit until core `n` is in reset.
7. Once the core is in reset, clear the `RCU_SIDIS.SIn` bit to re-enable the core interfaces.
8. Clear the `RCU_CRCTL.Rn` bit to take core `n` out of reset.
9. Poll the `RCU_CRSTAT.CRn` bit until core `n` is out of reset.

## BF60x Specific Information

The following information specific to BF60x processors should be kept in mind when reading this chapter:

- There are two cores, core 0 and core 1
- There is no RTC (real-time clock)
- The only SCLK0 domain source of system reset is the SDU
- There is no SCLK1 source of system reset
- The SYSCLK sources of system reset are TRU and SEC

## RCU Register Descriptions

Reset Control Unit (RCU) contains the following registers.

**Table 33-4:** RCU Module Registers

| Name          | Description                             |
|---------------|-----------------------------------------|
| RCU_CTL       | Control Register                        |
| RCU_STAT      | Status Register                         |
| RCU_CRCTL     | Core Reset Control Register             |
| RCU_CRSTAT    | Core Reset Status Register              |
| RCU_SIDIS     | System Interface Disable Register       |
| RCU_SISTAT    | System Interface Status Register        |
| RCU_SVECT_LCK | SVECT Lock Register                     |
| RCU_BCODE     | Boot Code Register                      |
| RCU_SVECTn    | Software Vector Register <code>n</code> |

## Control Register

The RCU control register (RCU\_CTL) provides a register lock, controls for the system reset pin, and a control for system reset.

**RCU\_CTL: Control Register - R/W**

Reset = 0x0000 0000

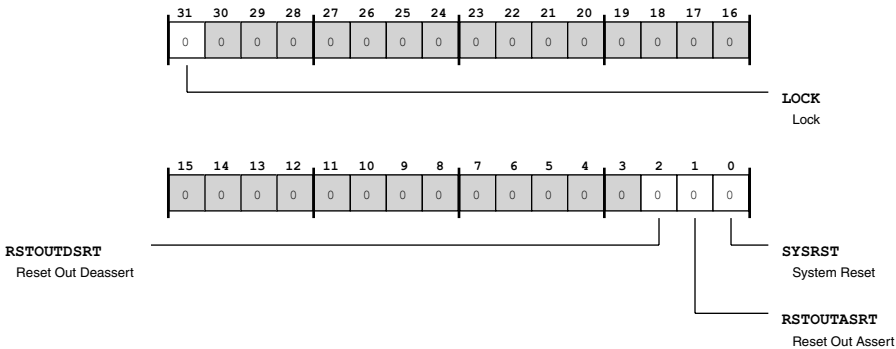


**Figure 33-1:** RCU\_CTL Register Diagram

**Table 33-5:** RCU\_CTL Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                                                                                                     |                 |
|---------|------------|---------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| 31      | LOCK       | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_CTL . LOCK bit is set, the RCU_CTL register is read only (locked). |                 |
|         |            | 0                                                                                                                                           | Unlock          |
|         |            | 1                                                                                                                                           | Lock            |
| 2       | RSTOUTDSRT | Reset Out Deassert.<br>The RCU_CTL . RSTOUTDSRT bit controls deassertion of the system reset pin.                                           |                 |
|         |            | 0                                                                                                                                           | No Action       |
|         |            | 1                                                                                                                                           | Deassert RSTOUT |

**Table 33-5:** RCU\_CTL Register Fields (Continued)

| Bit No. | Bit Name   | Description/Enumeration                                                                     |
|---------|------------|---------------------------------------------------------------------------------------------|
| 1       | RSTOUTASRT | Reset Out Assert.<br>The RCU_CTL.RSTOUTASRT bit controls assertion of the system reset pin. |
|         |            | 0 No Action                                                                                 |
|         |            | 1 Assert RSTOUT                                                                             |
| 0       | SYSRST     | System Reset.<br>The RCU_CTL.SYSRST bit provides reset for all system units.                |
|         |            | 0 No Action                                                                                 |
|         |            | 1 System Reset                                                                              |

## Status Register

The RCU status register (RCU\_STAT) contains status bits for all RCU reset sources, reset status, and boot mode inputs. Status bits for reset sources are sticky and can be cleared by software. Error status bits are cleared by any reset event.

### RCU\_STAT: Status Register - R/W

Reset = 0x0000 0021

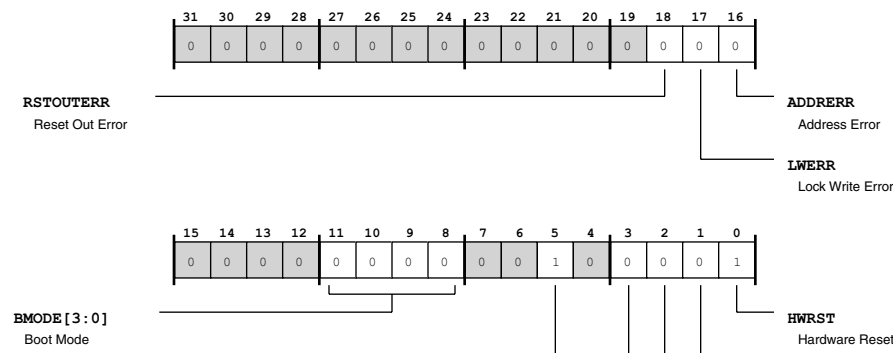


**Figure 33-2:** RCU\_STAT Register Diagram

**Table 33-6:** RCU\_STAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                       |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | RSTOUTERR | Reset Out Error.<br>The RCU_STAT.RSTOUTERR bit indicates (if set) that a write attempted to set the RCU_CTL.RSTOUTASRT and RCU_CTL.RSTOUTDSRT simultaneously. This condition triggers a bus error.                                                            |
|         |           | 0 No Error                                                                                                                                                                                                                                                    |
|         |           | 1 Error Occurred                                                                                                                                                                                                                                              |
| 17      | LWERR     | Lock Write Error.<br>The RCU_STAT.LWERR bit indicates (when set) there was an attempted write to an RCU register while the the RCU_CTL.LOCK bit was set and the global lock bit is enabled (SPU_CTL_GLCK bit =1). This status bit is sticky; write-1-to-clear |
|         |           | 0 No Error                                                                                                                                                                                                                                                    |
|         |           | 1 Error Occurred                                                                                                                                                                                                                                              |
| 16      | ADDRERR   | Address Error.<br>The RCU_STAT.ADDRERR bit indicates that the RCU generated an address error. This status bit is sticky; write-1-to-clear it.                                                                                                                 |
|         |           | 0 No Error                                                                                                                                                                                                                                                    |
|         |           | 1 Error Occurred                                                                                                                                                                                                                                              |
| 11:8    | BMODE     | Boot Mode.<br>The RCU_STAT.BMODE bits indicate the input on the boot mode pins.                                                                                                                                                                               |
| 5       | RSTOUT    | Reset Out Status.<br>The RCU_STAT.RSTOUT bit indicates the assertion status of the system reset pin.                                                                                                                                                          |
|         |           | 0 RSTOUT Deasserted                                                                                                                                                                                                                                           |
|         |           | 1 RSTOUT Asserted                                                                                                                                                                                                                                             |

**Table 33-6:** RCU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | SWRST    | Software Reset.<br>The RCU_STAT.SWRST bit indicates that a system reset (which was triggered by software) has occurred since the last time a hardware reset occurred or since the RCU_STAT.SWRST bit was cleared by software.                                                        |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |
| 2       | SSRST    | System Source Reset.<br>The RCU_STAT.SSRST bit indicates that a system reset triggered by hardware in the system clock domain, clock A domain, or clock B domain has occurred since the last time a hardware reset occurred or since the RCU_STAT.SSRST bit was cleared by software. |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |
| 1       | HBRST    | Hibernate Reset.<br>The RCU_STAT.HBRST bit indicates that a hibernate reset has occurred since the last time a hardware reset occurred or since the RCU_STAT.HBRST bit was cleared by software.                                                                                      |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |
| 0       | HWRST    | Hardware Reset.<br>The RCU_STAT.HWRST bit indicates that a hardware reset has occurred.                                                                                                                                                                                              |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |

## Core Reset Control Register

The RCU core reset control n registers (RCU\_CRCTL) include a lock bit (RCU\_CRCTL.LOCK) and a core reset bit (RCU\_CRCTL.CRn) for each processor core on the product.

RCU\_CRCTL: Core Reset Control Register - R/W

Reset = 0x0000 0002

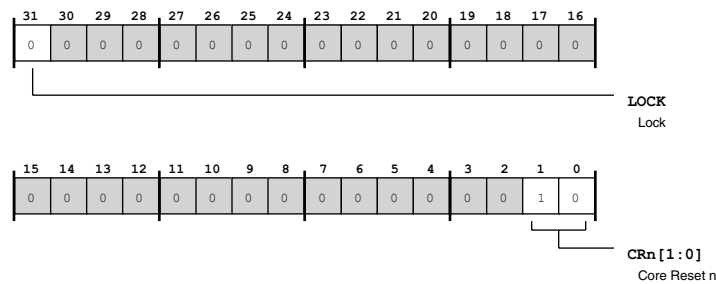


Figure 33-3: RCU\_CRCTL Register Diagram

Table 33-7: RCU\_CRCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                         |                        |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_CRCTL . LOCK bit is set, the RCU_CRCTL register is read only (locked). |                        |
|         |          | 0                                                                                                                                               | Unlock                 |
|         |          | 1                                                                                                                                               | Lock                   |
| 1:0     | CRn      | Core Reset n.<br>The RCU_CRCTL . CRn bit resets processor core n.                                                                               |                        |
|         |          | 0                                                                                                                                               | Inactive               |
|         |          | 1                                                                                                                                               | Reset Processor Core n |

Core Reset Status Register

The RCU core reset status register (RCU\_CRSTAT) contains status bits, indicating which core or cores have been reset.

RCU\_CRSTAT: Core Reset Status Register - R/W

Reset = 0x0000 0003

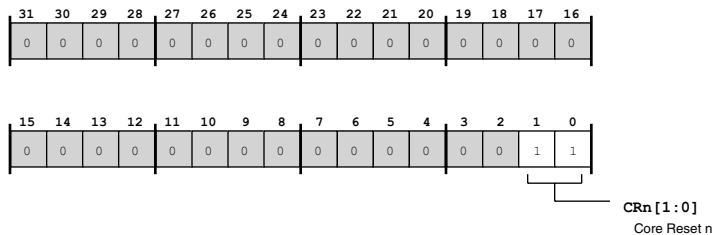


Figure 33-4: RCU\_CRSTAT Register Diagram

Table 33-8: RCU\_CRSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------|
| 1:0     | CRn      | Core Reset n.<br>The RCU_CRSTAT.CRn bits indicate which cores have been reset since the last time the bit was cleared. |
|         |          | 0 Inactive                                                                                                             |
|         |          | 1 Reset of Core n Occurred                                                                                             |

System Interface Disable Register

The RCU system interface disable register (RCU\_SIDIS lets the RCU selectively disable the functional units in the processor. For information on mapping between RCU\_SIDIS bits and functional units, see the RCU functional description.

RCU\_SIDIS: System Interface Disable Register - R/W

Reset = 0x0000 0000

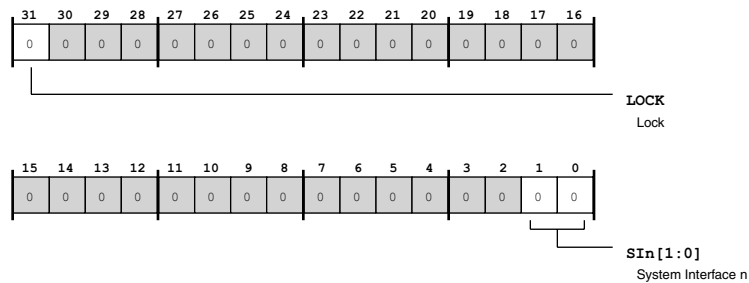


Figure 33-5: RCU\_SIDIS Register Diagram

Table 33-9: RCU\_SIDIS Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_SIDIS.LOCK bit is set, the RCU_SIDIS register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                 |
|         |          | 1                      Lock                                                                                                                   |
| 1:0     | SIn      | System Interface n.<br>The RCU_SIDIS.SIn bits select functional units in the processor to be disabled.                                        |
|         |          | 0                      Enable                                                                                                                 |
|         |          | 1                      Disable Functional Unit n                                                                                              |

System Interface Status Register

The RCU system interface status register (RCU\_SISTAT) indicates whether a functional unit has or has not acknowledged an RCU unit disable request.

RCU\_SISTAT: System Interface Status Register - R/NW

Reset = 0x0000 0000

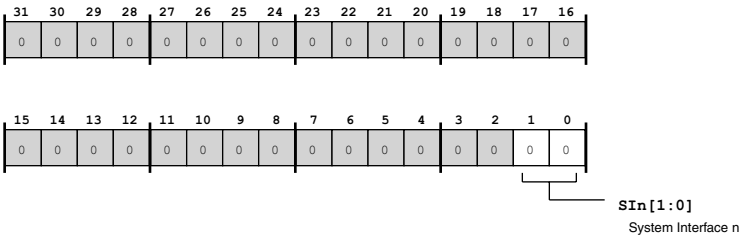


Figure 33-6: RCU\_SISTAT Register Diagram



Table 33-10: RCU\_SISTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration |                              |
|---------|----------|-------------------------|------------------------------|
| 1:0     | SIn      | System Interface n.     |                              |
|         |          | 0                       | No Acknowledge               |
|         |          | 1                       | Disable Request Acknowledged |

SVECT Lock Register

The RCU software vector lock register (RCU\_SVECT\_LCK) provides a register lock and software vector n enable bits for each processor core on the product.

RCU\_SVECT\_LCK: SVECT Lock Register - R/W

Reset = 0x0000 0000

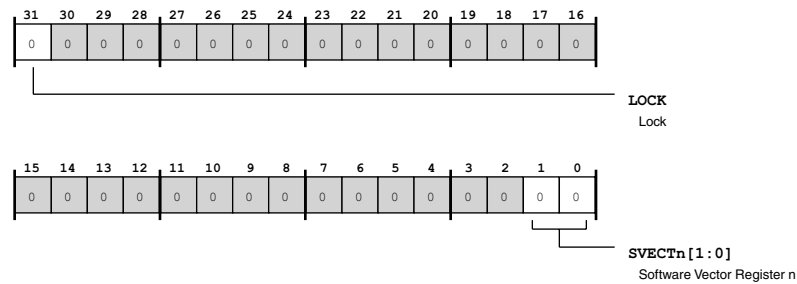


Figure 33-7: RCU\_SVECT\_LCK Register Diagram

Table 33-11: RCU\_SVECT\_LCK Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                               |        |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_SVECT_LCK.LOCK bit is set, the RCU_SVECT_LCK register is read only (locked). |        |
|         |          | 0                                                                                                                                                     | Unlock |
|         |          | 1                                                                                                                                                     | Lock   |

Table 33-11: RCU\_SVECT\_LCK Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                               |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------|
| 1:0     | SVECTn   | Software Vector Register n.<br>The RCU_SVECT_LCK.SVECTn bits enable a software vector (reset vector) for each core n. |
|         |          | 0                      Disable                                                                                        |
|         |          | 1                      Enable Software Vector for Core n                                                              |

Boot Code Register

The RCU software vector lock register (RCU\_BCODE) provides a register lock and software vector n enable bits for each processor core on the product.

RCU\_BCODE: Boot Code Register - R/W

Reset = 0x0000 0000

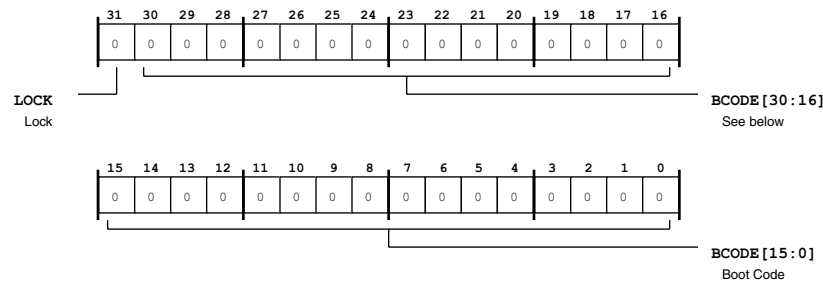


Figure 33-8: RCU\_BCODE Register Diagram

Table 33-12: RCU\_BCODE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_BCODE.LOCK bit is set, the RCU_BCODE register is read only (locked). |
|         |          | 0                      Unlock                                                                                                                 |
|         |          | 1                      Lock                                                                                                                   |

**Table 33-12:** RCU\_BCODE Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------|
| 30:0    | BCODE    | Boot Code.<br>The RCU_BCODE.BCODE bits contain a boot code for the processor. For more information, see the RCU functional description. |

## Software Vector Register n

The RCU software vector n registers (RCU\_SVECTn) each contain a software vector (reset vector) for the processor cores.

**RCU\_SVECTn: Software Vector Register n - R/W**

Reset = 0x0000 ffa0

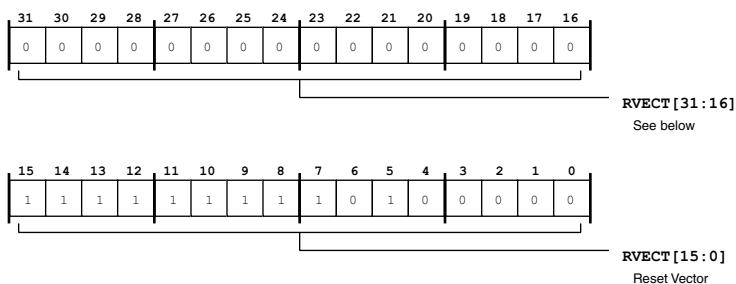


**Figure 33-9:** RCU\_SVECTn Register Diagram

**Table 33-13:** RCU\_SVECTn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | RVECT    | Reset Vector.           |



# 34 Booting the Processor (Boot)

This section provides an overview of the process of booting. **Notice of Changes:** This documentation contains material that is subject to change without notice. The content of the boot ROM as well as hardware behavior may change across silicon revisions. See the anomaly list for differences between silicon revisions.

When the `RESET` input signal releases, the processor starts fetching and executing from the on-chip boot ROM from the bottom of L2 ROM.

The internal boot ROM includes a boot kernel that loads the processor according to a boot stream provided in external memory or via a supported peripheral. A boot stream is generally produced using the loader utility provided with the tools. The boot stream consists of a series of blocks that can instruct the boot kernel to perform the functions described in this chapter, including:

- Loading memory with provided data or code
- Initializing memory
- Initializing peripherals
- Verifying checksum
- Decompress data
- Forward data to a peripheral

**NOTE:** All available peripherals are not supported by the boot kernel, for more information, see specific boot modes

The boot stream is processed in sequence until a block marked as the final block is processed. After the final block is processed the boot kernel jumps to the start of the program location. An application generally starts at address `0xFFA0000` unless otherwise specified.

The source of the boot kernel is determined by which boot mode is selected. A boot mode is specified by setting the four dedicated `BMODE` input pins.

## Boot Loader Stream

A loader stream is a set of formatted blocks containing instructions for the boot kernel, as well as the application and data to be loaded to the chip. This section describes in detail different aspects of the stream, its blocks, and some common use cases.

Each block begins with a block header which contains attributes of the block as well as flags to control its processing by the boot ROM. On power-up or reset the processor begins executing the on-chip boot ROM and the boot stream is either read from memory or received from a peripheral, depending on the boot mode specified. Each block in the boot stream instructs the boot kernel to perform some action, most commonly to simply load data to a specified location. Other actions include running code that initializes a peripheral, forwarding data to a peripheral, or processing data then loading it to a location.

A loader stream must always begin with a First block, and end with a Final block.

As the Project Flow figure illustrates, the tools suite includes a loader utility (loader.exe). The loader utility parses an input executable file (.dxe), and creates a set of blocks representing the segmented application. The output is stored in a loader file (.ldr). The loader file contains the boot stream and is made available to hardware by programming or burning it into non-volatile external memory, or sending it through a peripheral during boot time. Refer to the Loader Manual for details on loader usage.

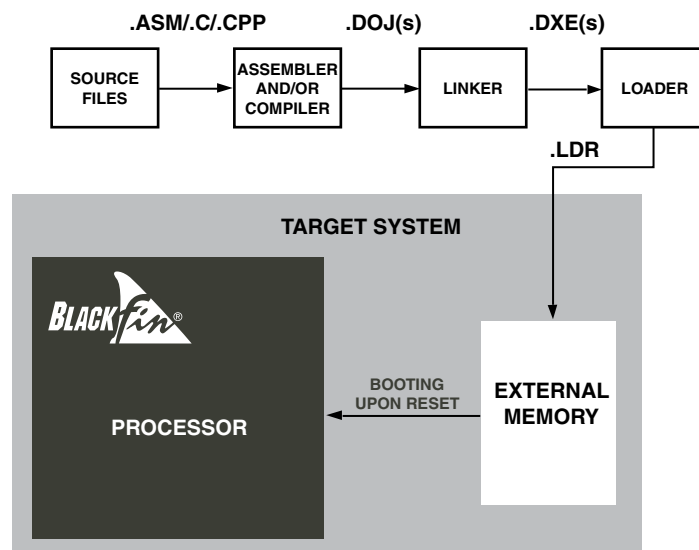


**Figure 34-1:** Project Flow

The [Booting Process](#) figure shows the parallel or serial boot stream contained in a flash memory device. In host boot scenarios, the non-volatile memory usually connects to the host processor rather than directly to the processor. After reset the headers are read and parsed by the on-chip boot kernel and the loader stream is processed block by block. Finally, payload data is copied to destination addresses, either in on-chip L1 and L2 memory, or off-chip to SDRAM or SRAM.

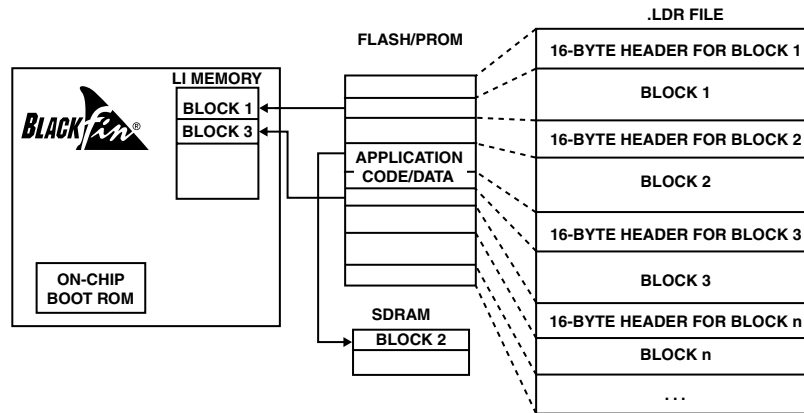


**Figure 34-2:** Booting Process

In some cases (for example when the BLFAG\_INDIRECT flag for any block is set), the boot kernel uses another memory block in L1 data bank B for intermediate data storage. To avoid conflicts, the elfloader utility ensures this region is booted last.

The entire source code of the boot ROM is shipped with the tools. The source code can be reviewed for information not covered in this manual. Note that minor maintenance work may be required to the boot ROM code if silicon is updated.

## Block Structure

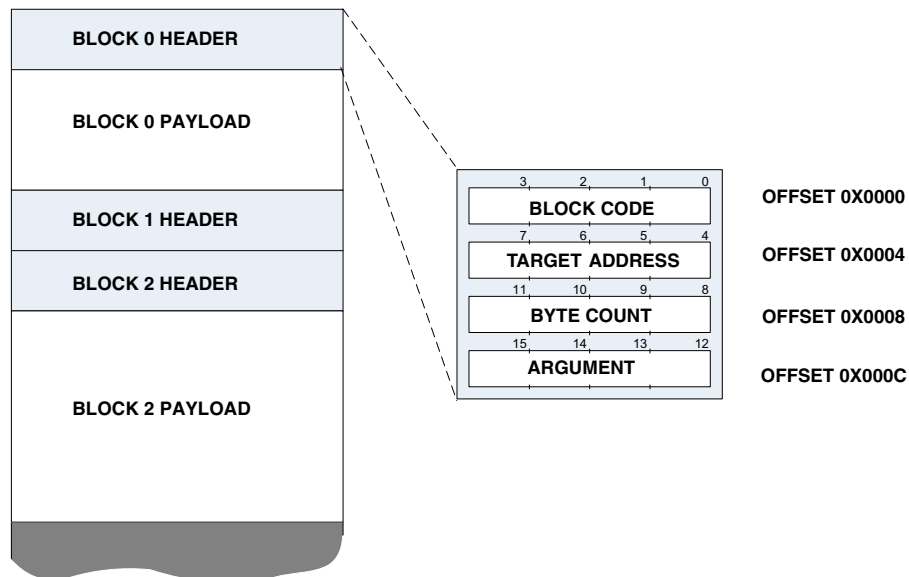


**Figure 34-3:** Boot Header

A boot stream consists of multiple boot blocks as shown in the figure. Every block is headed by a 16-byte block header. The 16 bytes are functionally grouped into four 32-bit words: the block code, target address, byte count, and the argument field. This section describes the fields in general. The uses may vary

depending on the particular block type and boot mode, refer to the block type descriptions and boot modes for further information.

## Block Code

**Table 34-1:** Block Header flags

| Bit   | Name            | Description                                                                                                                                          |
|-------|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0-3   | BCODE           | Specific to boot modes (see Boot Modes)                                                                                                              |
| 4     | BFLAG_SAVE      | Saves the memory of this block to off-chip memory in case of power failure or a hibernate request. This flag is not used by the on-chip boot kernel. |
| 5     | BFLAG_AUX       | Nests special block types as required by special-purpose second-stage loaders. This flag is not used by the on-chip boot kernel.                     |
| 6     | reserved        |                                                                                                                                                      |
| 7     | BFLAG_FORWARD   | Forward payload to a device. Must be used in conjunction with the BFLAG_INDIRECT flag.                                                               |
| 8     | BFLAG_FILL      | Fill the target location with a specified value.                                                                                                     |
| 9     | BFLAG_QUICKBOOT | Does not process block for a quick boot (warm boot).                                                                                                 |
| 10    | BFLAG_CALLBACK  | Calls function at the address provided.                                                                                                              |
| 11    | BFLAG_INIT      | Calls function at target address after loading payload to the same address                                                                           |
| 12    | BFLAG_IGNORE    | Block is ignored.                                                                                                                                    |
| 13    | BFLAG_INDIRECT  | Boots to an intermediate storage place.                                                                                                              |
| 14    | BFLAG_FIRST     | Indicates the block to be the first block of a new dxs                                                                                               |
| 15    | BFLAG_FINAL     | Indicates the last block of a loader stream. Booting will complete after processing the block.                                                       |
| 16-23 | HDRCHK          | A simple XOR checksum of the other 31 bytes in the boot block header.                                                                                |
| 24-31 | HDRSIGN         | 0xAD (dependent on processor family)                                                                                                                 |



## TARGET\_ADDRESS

The `TARGET_ADDRESS` holds the address that the block applies to, (where the code or data should be loaded). However, the field is interpreted differently depending on what specific flags are set in the block code. Refer to each Block Type's documentation for details.

The following attributes must be true:

- The target address must be divisible by 4, as the boot kernel uses 32-bit DMA for certain operations.
- The target address must point to valid on-chip or off-chip memory locations.
- The `BFLAG_INDIRECT` flag must be set if the target address points to L1 instruction memory. For performance reasons, this is also recommended when booting to off-chip memory.

**NOTE:** Booting into scratchpad memory is not supported. If booting to scratchpad memory is attempted, the processor hangs within the on-chip boot ROM. Similarly, booting into the upper 16 bytes of L1 data bank A is not supported. These memory locations are used by the kernel for intermediate storage of block header information. These memory regions cannot be initialized at boot time. After booting, they can be used by the application during runtime.

## BYTE\_COUNT

The byte count must be divisible by 4, and also may be zero. This 32-bit field generally holds the size of the block. In some cases it can be used differently (such as when `BFLAG_FILL` is set). See the block types section for information on specific variations.

## ARGUMENT

The 32-bit field is a user variable for most block types. The value is accessible by the initcode or the callback routine and can therefore be used for optional instructions to these routines.

The `ARGUMENT` field is used in different ways by different block types. See the block type descriptions for further information.

## Block Types

Describes how to use different types of loader blocks.

A loader stream is a set of linked blocks and each block is responsible for performing a certain function dependent on the block's type. A block type is defined by its flags in the block header. Operations include functions such as loading data, filling a memory region with data, forwarding data to a peripheral, and instructing the kernel to stop processing. This section describes each block type and how it is used within a boot loader stream.

## Normal Block

A block's primary function is to load some data into a specified location of memory. A normal block instructs the boot kernel to load the data contained in its payload to the location specified in the `TARGET_ADDRESS` field. The `BYTE_COUNT` defines the size of the payload, once the correct amount of data has been loaded, the kernel moves on to process the next block in the stream.

**Table 34-2:** Flags

| Flag                        | Required Value |                                                 |
|-----------------------------|----------------|-------------------------------------------------|
| <code>TARGET_ADDRESS</code> | Y              | Address where payload is loaded (must be valid) |
| <code>BYTE_COUNT</code>     | Y              | Size of block in bytes                          |

## First Block

A first block indicates the start of a DXE and is always required at the beginning of a loader stream. In the case of a loader stream that contains *Multi-DXE Boot Streams*, a first block occurring within the loader indicates the beginning of a new DXE.

When the kernel processes the first block in a loader stream, the `TARGET_ADDRESS` also updates the `EVT1` register. This register contains the start address of the application.

**NOTE:** Note that a First Block cannot be combined with a Fill Block

**Table 34-3:** Flags

| Flag                        | Required Value |                                                                                                                                                                                                               |
|-----------------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>BFLAG_FIRST</code>    | Y              | 1                                                                                                                                                                                                             |
| <code>ARGUMENT</code>       | Y              | Offset to the next DXE, or first address following loader stream                                                                                                                                              |
| <code>TARGET_ADDRESS</code> | Y              | When the block is the first block in a loader stream, also defines the start address for the application. If the block is not the first in a loader stream the target address is used as in normal operation. |

## Final Block

The final block marks the last block in a boot stream (not a DXE). After processing a final block the boot kernel jumps to the application's start address. By default this is performed by jumping to the vector stored in the `EVT1` register. The boot kernel provides options to execute an `RTS` instruction or a `RAISE 1` instruc-

tion. The default behavior can be changed by an initcode routine, or by the kernel's boot routine API. The EVT1 register is updated by the boot kernel when processing the first block in the loader stream.

Before the boot kernel passes program control to the application it does some housekeeping. Most of the registers that were used are put in their default state. However, some register values may differ depending on the boot mode. See Boot Modes for more information. The following DMA configuration registers and primary register control registers (UART\_CTL, SPI\_CTL, HOST\_CONTROL for example) are restored, while others are purposely not restored. For example SPI\_CLK.BAUD and UART\_CLK remain unchanged so that the settings obtained during the boot process are not lost.

**Table 34-4:** Flags

| Flag        | Required Value |   |
|-------------|----------------|---|
| BFLAG_FINAL | Y              | 1 |

## Indirect Block

An intermediate block is first loaded to a storage location before being copied to the destination. This functionality is motivated by the following situations:

- Some boot modes may not use DMA. The core cannot access some memory locations directly (L1 instruction SRAM), and some it cannot access efficiently. An intermediate load to a different location improves overall efficiency.
- In some booting scenarios the data in the payload needs to be operated on or analyzed before it is fully loaded (such as decryption or checksum calculation). By using an intermediate location such scenarios are simplified and can be more efficient when loading to off-chip memories (see Callback Block).

In some cases a boot block may not fit into temporary storage memory so having a larger buffer may improve boot performance. If an entire block cannot fit into the buffer it is processed in pieces. Initialization code or callback functions can alter the temporary buffer region, including its location and size, by modifying the pTempBuffer and dTempByteCount variables in the [STRUCT\\_ROM\\_BOOT\\_CONFIG](#) structure. The default region is at the upper end of a physical memory block. When the size is changed from the default, keep in mind that it is already at the end of a memory block, so the beginning address pTempBuffer must also be modified.

**Table 34-5:** Flags

| Flag           | Required Value |                                                                                                     |
|----------------|----------------|-----------------------------------------------------------------------------------------------------|
| BFLAG_INDIRECT | Y              | 1                                                                                                   |
| BFLAG_CALLBACK | N              | Defines a callback function to operate on intermediate data. These 2 flags are often used together. |

## Ignore Block

An ignore block is a block that is (in most cases) ignored by the loader stream. Ignore blocks are useful when it's not possible to pass information in another block header. For example, if the first block contains data rather than application code, then inserting a first block with the correct application start address ensures the correct start address is used. Since this block has no other function it should be marked as an ignore block so that the kernel will not attempt to process any payload.

**Table 34-6:** Flags

| Flag         | Required Value |                                      |
|--------------|----------------|--------------------------------------|
| BFLAG_IGNORE | Y              | 1                                    |
| BYTE_COUNT   | Y              | Size of block to ignore, may be zero |

## Init Block

An init (initialization) block instructs the boot kernel to issue a `CALL` instruction to the target address after the entire block has been loaded. The function called is referred to as the *initcode routine*. If the initcode routine has been previously loaded, the block may declare a zero-size and have no payload.

Initcode routines can be used to speed up and customize booting mechanisms exposed by the boot kernel. Traditionally, an initcode routine is used to setup system PLL, bit rates, wait states, and the SDRAM controller. If executed early in the boot process, the boot time can be significantly reduced.

Initcode routines are required to follow the C language calling conventions. The expected C prototype is:

```
void initcode(STRUCT_ROM_BOOT_CONFIG* pBootStruct)
```

When programming in assembly, be certain to return using an `RTS` instruction. See the *C/C++ Compiler and Library Manual* for more information.

The struct provided to the initcode routine by the boot kernel contains a variety of information about the block being processed. This includes header information, locations of temporary block data (for indirect blocks), target address, and byte count. See the

In the simplest case, an initcode routine consists of only a single block in which the `BFLAG_INIT` flag is set. For larger routines, a sequence of blocks can incrementally load the routine, and only the last block should set the `BFLAG_INIT` flag. In the latter case, the last block should be of size zero, and simply instruct the boot kernel to issue the `CALL` instruction.

An initcode routine can be overwritten by a successive block if it is no longer needed, otherwise the routine can be called at multiple points during the boot process, and even remain in memory after booting is completed for use by the application.

**NOTE:** The following list provides requirements for initcode that is written in C or C++.

- Ensure the initcode routine does not contain calls to the run-time libraries
- Do not assume that parts of the run-time environment, such as the heap, are fully functional
- Ensure that all run-time components are loaded and initialized before the routine executes

The *loader* utility and tool set provide mechanisms to aid in implementing init codes and organizing them properly within the boot loader stream. A special project type is provided to allow the creation of initcode routines as separate projects. Options are available to assign particular pieces of the application to be initcode routines. For details and more information on the utility, see the *Loader Utility Manual*.

Initcode routine examples, including SDRAM controller initialization, can be found in the examples section

**Table 34-7:** Flags

| Flag           | Required Value |                                                                              |
|----------------|----------------|------------------------------------------------------------------------------|
| BFLAG_INIT     | Y              | 1                                                                            |
| TARGET_ADDRESS | Y              | Location to load payload data. CALL instruction issued to the same location. |
| ARGUMENT       | N              | Can be used to supply block specific arguments                               |
| BYTE_COUNT     | Y              | Size of payload, may be zero                                                 |

## Callback Block

A callback block instructs the boot kernel to call a pre-registered function upon completion of loading the block's payload. The purpose of a callback routine is to apply standard processing to the block payload. The callback routine is registered through an initcode routine prior to loading a block using the routine. Typically, callback routines contain checksum, decryption, decompression or hash algorithms.

To register a callback an init block must be created whose initcode modifies the `pCallbackFunction` pointer in the `STRUCT_ROM_BOOT_CONFIG` structure. A callback routine must be registered before a callback block can be processed.

Since callback routines require access to the payload data of the boot blocks, the block data must be loaded before it can be processed. Often an INDIRECT block is used in combination with a callback block.

Callback routines are expected to meet the C language calling conventions. For more information see the *C/C++ Compiler and Library Manual*. The prototype is as follows: `int32_t CallbackFunction(struct STRUCT_ROM_BOOT_CONFIG* pBootStruct, ADI_BOOT_BUFFER* pCallbackStruct, int32_t dCbFlags)`

The `pBootStruct` argument contains the `STRUCT_ROM_BOOT_CONFIG` information, and the `pCallbackStruct` contains the target address and size of the block (may vary when using indirect). The `dCbFlags` parameter is specifically used when indirect is also used. The `CBFLAG_DIRECT` flag indicates that the `BFLAG_INDIRECT` bit is not active and so that the callback routine is only called once per block. When the `CBFLAG_DIRECT` is set, the `CBFLAG_FIRST` and `CBFLAG_FINAL` are also set.

```
#define CBFLAG_FINAL 0x8  #define CBFLAG_FIRST 0x4  #define CBFLAG_DIRECT 0x1
```

See the Data Structures section for more detailed information on these data.

Callback Block Used in Conjunction with Indirect Block

When a block using a callback routine is also loaded indirectly there are slight behavior differences. The procedure for loading is:

- 1. Data is loaded into the temporary buffer defined by the `pTempBuffer`.
- 2. The `CALL` to the `pCallBackFunction` is issued.
- 3. After the callback routine returns, if the return value is zero, the memory DMA copies data to the destination

If a block does not fit entirely into the temporary buffer, loading is performed similar to indirect blocks, and the callback function is called after each chunk is loaded into the temporary storage. The `dCbFlags` parameter gives information on the specific iteration.

When a block does not fit entirely into the temporary storage area, the `dCbFlags` tells the callback routine whether it is invoked for the first time (`CBFLAG_FIRST`) or whether it is called the last time (`CBFLAG_FINAL`) for a specific block.

When DMA is invoked to copy the data, it relies on the `pCallbackStruct` structure, not the global `pTempBuffer` and `dTempByteCount` variables. The callback routine can control the source of the memory DMA by altering the content of the `pCallbackStruct` structure, as may be required if the callback routine performs data manipulation such as decompression.

When an indirect block is used, the return value of the callback routine determines whether the DMA transfer occurs. If the value is non-zero, then the transfer does not occur.

Table 34-8: Flags

| Flag           | Required Value |   |
|----------------|----------------|---|
| BFLAG_CALLBACK | Y              | 1 |

## Quick Boot Block

Quick Blocks are only processed for a full boot. In some booting scenarios, not all memories need to be re-initialized. For example, in a wake-up from hibernate state, off-chip SRAM may not be impacted if it is powered while the processor is in a hibernate state. Dynamic RAM may also not be impacted if it was put into a self-refresh mode before the processor powered down.

The boot kernel uses the following parameters to determine whether or not a quick block should be processed. See the "Global Boot Flow" figure also.

- The `SYSCR` register is read to determine what kind of boot is expected from the boot kernel. The `WURESET` bit is compared against other reset bits to distinguish between cold boot and warm boot situations and to identify wake-up from hibernate situations.
- The `BCODE` bit field in the `SYSCR` register can overrule the native decision of the boot kernel for a software boot.
- The `BFLAG_WAKEUP` bit in the `dFlag` word of the `STRUCT_ROM_BOOT_CONFIG` structure indicates that the final decision was to perform a quick boot. If the boot kernel is called from the application, then the application can control the boot kernel behavior by setting the `BFLAG_WAKEUP` flag accordingly.
- The `BFLAG_QUICKBOOT` flag in the block code word of the block header controls whether the current block is ignored for quick boot.

**Table 34-9:** Flags

| Flag                         | Required Value |   |
|------------------------------|----------------|---|
| <code>BFLAG_QUICKBOOT</code> | Y              | 1 |

## Save Block

A save block saves the payload of the block to off-chip memory. This flag is not used by the on-chip boot kernel

**Table 34-10:** Flags

| Flag                    | Required Value |   |
|-------------------------|----------------|---|
| <code>BFLAG_SAVE</code> | Y              | 1 |

## Forward Block

A forward block is similar to a normal block except that the payload is written to a specified peripheral rather than loaded to memory. Rather than specifying a target address, a forward port is defined and the

data is forwarded accordingly. This feature can enable multiple processors in a system to be booted from the same source. This can be done in any hierarchical way using daisy chaining via supported peripherals. An entire boot stream can be contained in the payload of a block, and that entire stream can be forwarded to a peripheral.

The processor supports forwarding to the SPI and link ports. Any peripheral must first be initialized before any forward block that is using that peripheral is processed. This is accomplished by using the boot kernel's *Forward Config* (FUNC\_ROM\_FWDCFG) function. See the ROM API for more information on this function. The forward configuration function takes as arguments the specific peripheral to be initialized and performs all the required initialization for that peripheral to enable forwarding.

As mentioned earlier, the target address for forward blocks is a forward port rather than target address. The details of this parameter are peripheral specific. For purposes of forwarding, the TARGET\_ADDRESS field in the block header is referred to as dForwardPort.

Forwarding to SPI

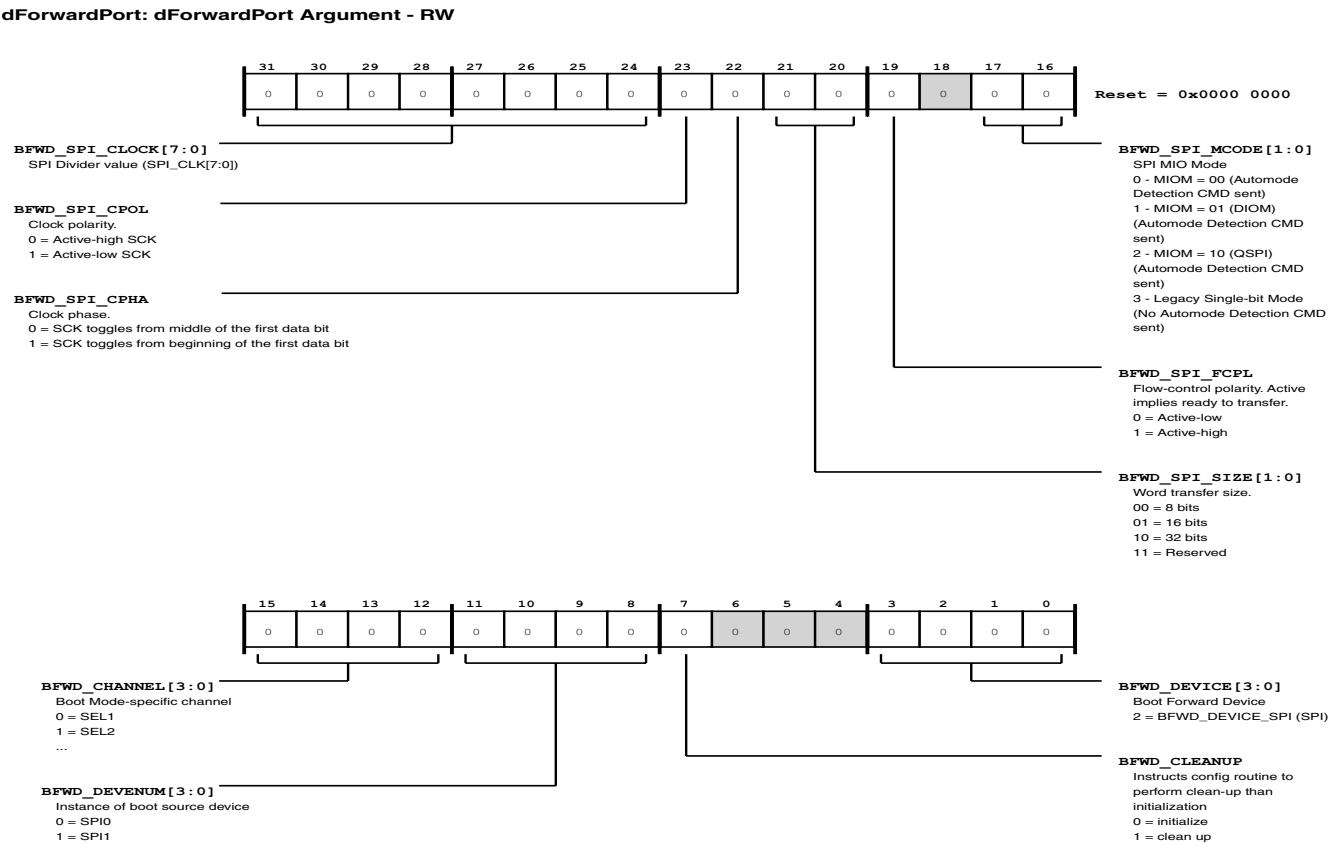


Figure 34-4: SPI dForward Port



The *SPI dForward Port* figure describes the various options available in the `dForwardPort` field for SPI block forwarding.

- For forwarding a boot stream to the SPI port, the `BFWD_DEVICE` field in the `dForwardPort` argument needs to be set to `BFWD_DEVICE_SPI`. The device source can be SPI0 or SPI1, depending on the `BFWD_DEVENUM` field.
- When using both device sources, the Forward Config API routine needs to be called twice.
- A source device can serve multiple SPI slave devices and is selectable using the SPI SSELx signal (set in `BFWD_CHANNEL`). If multiple slave devices are connected to the FCS signal (`SPI_RDY`), a TBD pull-down resistor is absolutely required as the SPI slave devices have the `SPI_CTL.ODM` bit set to avoid conflicts on the SPI bus. The `BFWD_SPI_FCPL` register must be TBD (set/cleared).

The `BFWD_SPI_MCODE` field is used to choose the I/O mode; 0, 1, or 2 choose single-bit, dual-bit, or quad-bit mode. Setting `BFWD_SPI_MCODE` to 0, 1, or 2 also enables the automode detection command in front of the boot stream. See SPI Slave Boot specification for further details.

The `BFWD_SPI_MCODE3` register selects the legacy single-bit mode without sending any automode detection command. The `BFWD_SPI_CLOCK` register is required to set the lower 8 bits of the BAUD field in the `SPI_CLOCK` divider register. The upper 8 bits of the BAUD field are set to 0. The other fields are used to directly manipulate specific SPI Control register settings.

## Forwarding to the Link Ports

`dForwardPort`: `dForwardPort` Argument - RW

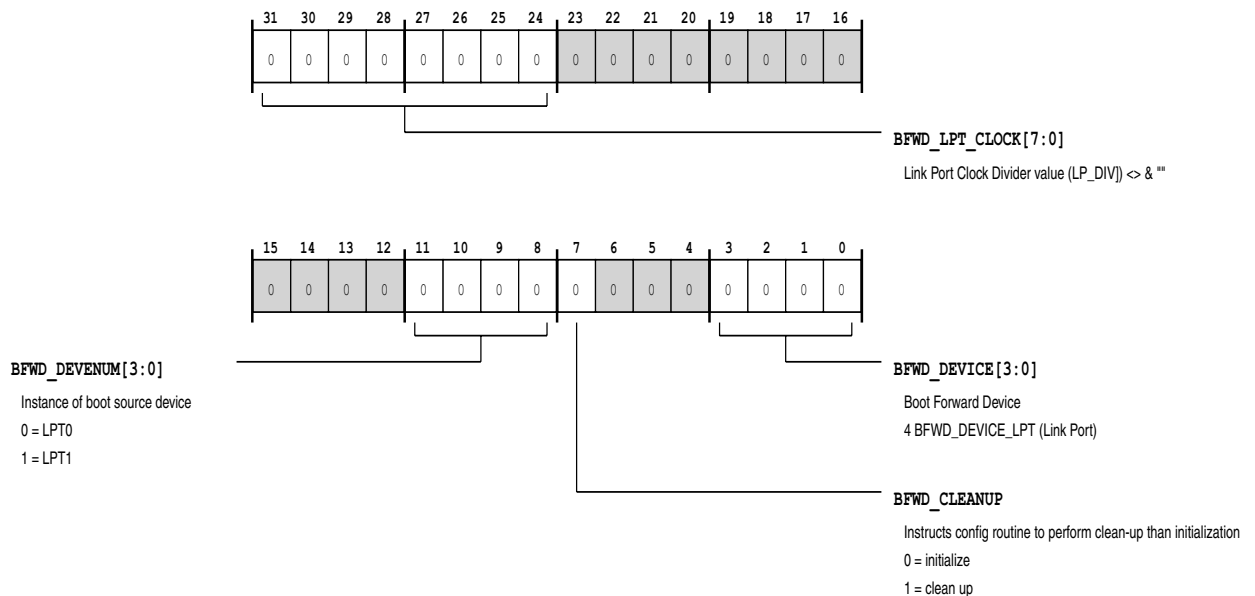


**Figure 34-5:** Link Port `dForward Port`

The link port dForward port describes the various options available in the dForwardPort field for link port block forwarding. For forwarding data to any of the link ports, the BFWD\_DEVICE field in the dForwardPort argument needs to be set to BFWD\_DEVICE\_LPT. The BFWD\_DEVENUM field selects whether data is forwarded to LPT0, LPT1, LPT2 or LPT3. Data can be forwarded to multiple link ports. However, the Forward Config routine needs to be called for each link port separately. The only programming option for Forwarding through link ports is the BFWD\_SPI\_CLOCK field in the dForwardPort argument. The Forward Config routines copies the value into the respective LPx\_DIV register. The Forward Callback routine does not evaluate this bit field. The Forward Config routine automatically sets the pin muxing as required and initializes the link port peripheral.

**Table 34-11:** Flags

| Flag           | Required Value |                                                            |
|----------------|----------------|------------------------------------------------------------|
| BFLAG_FORWARD  | Y              | 1                                                          |
| BFLAG_INDIRECT | Y              | 1                                                          |
| TARGET_ADDRESS | Y              | Contents of the peripheral specific dForwardPort parameter |

## Multi-DXE Boot Streams

This section describes Multi-DXE boot loader streams.

As seen by the elfloader, a boot stream is always generated from a DXE file. It is therefore common to refer to multi-DXE or multi-application booting. When the elfloader utility accepts multiple DXE files on its command line, it generates a contiguous boot image by default. The second boot stream is appended immediately to the first one. Since the utility updates the ARGUMENT field of all BFLAG\_FIRST blocks, the ARGUMENT field of a BFLAG\_FIRST block is called the next-DXE pointer.

The next-DXE pointer of the first DXE boot stream points relatively to the start address of the second DXE boot stream. A multi-DXE boot image can be seen as a linked list of boot streams. The next-DXE pointer of the last DXE boot stream points relatively to the next free address. This is illustrated by an example shown in the [Multi-DXE Boot Stream Example](#) figure.

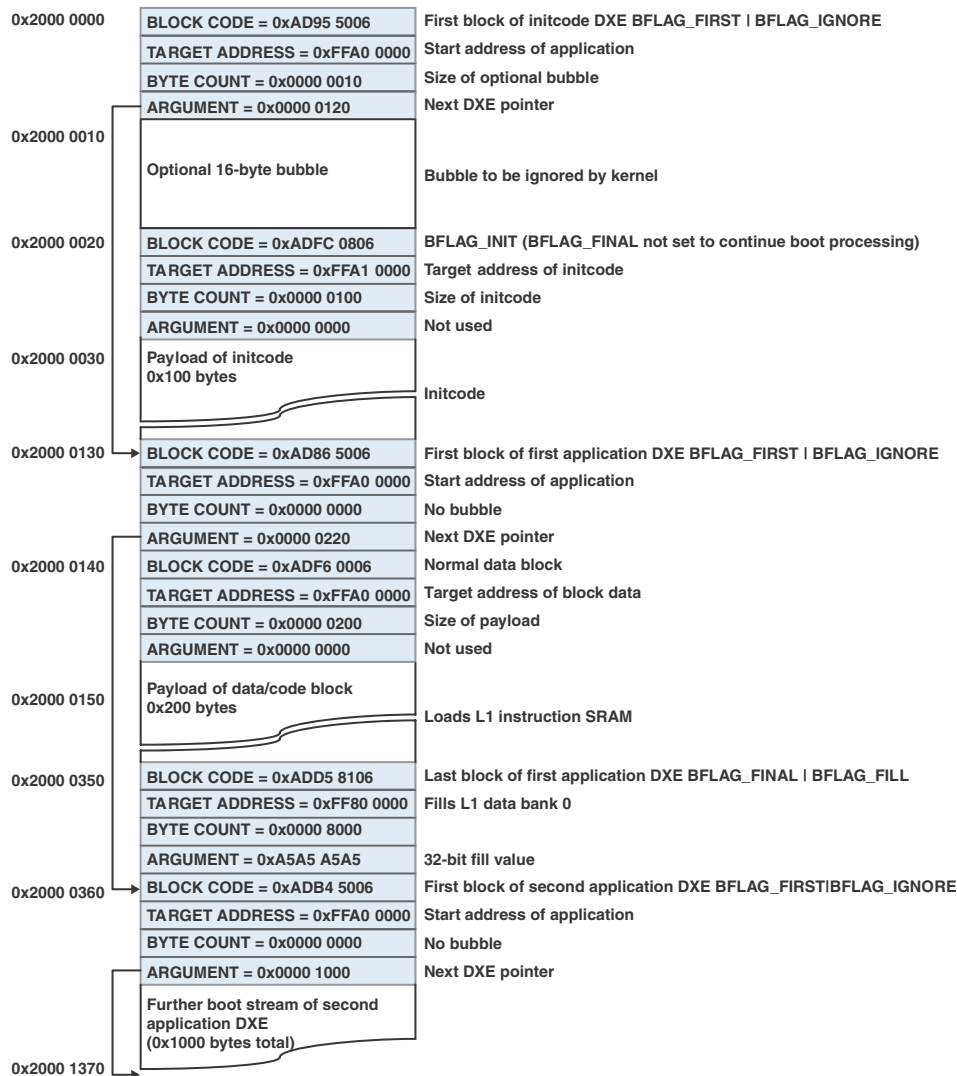


**Figure 34-6:** Multi-DXE Boot Stream Example

The *Multi-DXE Direct Code Execution Example* figure shows a linked list of initial block headers that instruct the boot kernel to terminate immediately and to start code execution at the address provided by the target address field of the individual blocks. There is nothing in the boot ROM that prevents multi-DXE applications from mixing regular boot streams and direct code execution blocks.

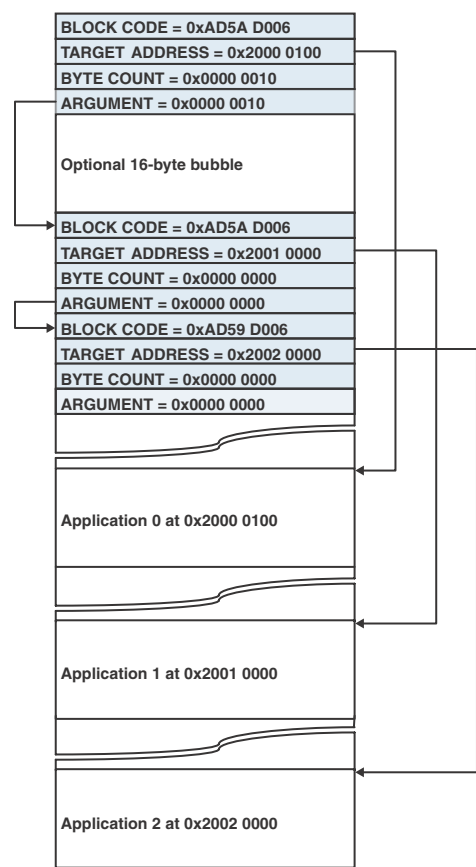


Figure 34-7: Multi-DXE Direct Code Execution Example

Single-Block Boot Streams

This section describes how to bypass booting and execute code directly from SDRAM.

The simplest boot stream consists of a single block header and one contiguous block of instructions. When the appropriate flags are set in the block header, the kernel loads the block to the target address, terminates, and begins executing from the target address of the block.

The *Initial Header for Single-Block Stream* table shows an example of a single-block boot stream header settings that can be loaded using any boot mode. The BFLAG\_FIRST and BFLAG\_FINAL flags are both set at the same time and the target address and byte count are determined by the desired location of the application.

Table 34-12: Initial Header for Single-Block Stream

| Field      | Description of Value                      |
|------------|-------------------------------------------|
| BLOCK_CODE | 0xAD000000 XORSUM BFLAG_FINAL BFLAG_FIRST |

**Table 34-12:** Initial Header for Single-Block Stream (Continued)

| Field          | Description of Value                                     |
|----------------|----------------------------------------------------------|
| TARGET_ADDRESS | Start address of block and application code              |
| BYTE_COUNT     | number of bytes in the block                             |
| ARGUMENT       | Functions as next-DXE pointer in multi-DXE boot streams. |

## Direct Code Execution

This section describes how to bypass booting and execute code directly from SDRAM.

Applications may want to avoid long booting times and start code execution directly from flash or SDRAM memory. This feature is called direct code execution.

An initial boot block header is required for the processor to fetch and execute program code from the boot device as early as possible. The safety mechanisms of the block, such as the header signature and the XOR checksum, are used to avoid unpredictable processor behavior when boot memory is not yet being programmed with valid data. Rather than blindly executing code, the boot kernel first executes the preboot routine for system customization, then loads the first block header and checks it for consistency. If the block header is corrupted, the boot kernel goes into a safe idle state and does not start code execution.

If the initial block header check is good, the boot kernel interrogates the block flags. If the block has the BFLAG\_FINAL flag set, the boot kernel immediately terminates and jumps directly to the address stored in the EVT1 register. To cause the boot kernel to customize the EVT1 register in advance, the first block must also have the BFLAG\_FIRST flag set. The target address field is then copied to the EVT1 register. In this way the target address field of the initial block defines the start address of the application.

For example, if BMODE = 1, when the block header described in the Initial Header table is placed at address 0x20000000, the boot kernel is instructed to issue a JUMP command to address 0x20000020.

The development tools must be instructed to link the above block to address 0x20000000 and the application code to address 0x20000020. An example shown in "Direct Code Execution" illustrates how this is accomplished using the tools suite.

**Table 34-13:** Initial Header

| Field          | Value      | Comments                                                                |
|----------------|------------|-------------------------------------------------------------------------|
| BLOCK_CODE     | 0xAD7BD006 | 0xAD000000 XORSUM BFLAG_FINAL BFLAG_FIRST BFLAG_IGNORE (MDMACODE & 0x6) |
| TARGET_ADDRESS | 0x20000020 | Start address of application code                                       |

**Table 34-13:** Initial Header (Continued)

| Field      | Value      | Comments                                                                                                                  |
|------------|------------|---------------------------------------------------------------------------------------------------------------------------|
| BYTE_COUNT | 0x00000010 | Ignores 16 bytes to provide space for control data such as version code and build data. This is optional and can be zero. |
| ARGUMENT   | 0x00000010 | Functions as next-DXE pointer in multi-DXE boot streams.                                                                  |

## Boot Modes

The boot kernel provides a variety of built-in support, such as call modes, for booting to various peripherals. The [Booting Modes](#) table describes the various boot modes.

In *slave* boot modes, the processor functions as a slave to any host device. In these modes the processor RESET input is usually controlled by the host device. Typically the host sets RESET and waits until the processor is ready to boot, depending on the peripheral being used, and transmits the boot stream data to the processor.

In a *master* boot mode, the processor controls the peripheral and indicates to the peripheral when to transmit the boot stream data.

**Table 34-14:** Booting Modes

| BMODE[2:0] | Boot Source      | Description                                                                          |
|------------|------------------|--------------------------------------------------------------------------------------|
| 000        | No Boot -idle    | The processor does not boot. Rather the boot kernel executes and IDLE instruction.   |
| 001        | Memory boot      | Boot from an address in memory, by default this is the start address of flash memory |
| 010        | RSI master boot  | Boot through the Removable Storage Interface (RSI) peripheral configure as a slave   |
| 011        | SPI0 master boot | Boot through the Serial Port Interface (SPI0) peripheral configured as a slave       |
| 100        | SPI0 slave boot  | Boot through the SPI0 peripheral configured as a master                              |
| 101        | reserved         | reserved                                                                             |
| 110        | LP0 slave boot   | Boot through the Link Port (LP0) peripheral configured as a master                   |
| 111        | UART0 slave boot | Boot through the UART0 peripheral configured as a master                             |

## No-Boot Mode

When the `BMODE` pins are all tied low (`BMODE=0000`), the processor does not boot. Instead it throws an emulation interrupt followed by an idle instruction. The purpose of this mode is to bring the processor up to a clean state from any unwanted state that may have been caused by incorrect programming of the boot source memory.

When connecting to an emulator and starting a debug session, the processor uses the emulation interrupt to wake up the processor and it can be debugged in the normal manner.

## Memory Boot Mode

The memory booting modes are intended for booting from flash, EEPROM, or any memory-mapped devices. Although this is a single `BMODE` setting, there are various configurations described in the MDMA Codes table.

By default the boot kernel does not alter any `EBIO` registers. Therefore, traditional asynchronous memory is assumed and maximum wait states are applied. Once the processor boots, the boot kernel loads an initial burst of 160-bit words, and interrogates the loaded `MDMACODE`. `MDMACODE` is the specialized version of the `BCODE` field defined in a generic block header.

The `MDMACODE` is filled by the `elfloader` utility based on boot mode, `-width` and `-dmawidth` settings. See the Loader and Utility Manual for more details.

**NOTE:** If the NOR flash device has been put into burst or page mode, it must be programmed back to the standard mode before the processor is reset. If the processor can reset itself without software control (via watchdog or double-fault error) a mechanism must be installed that also resets the flash device back to its default mode along with the processor.

Hardware configurations for the individual modes are shown in the 8/16-bit Flash Interconnection figure. The chip select is always controlled by the `ASMO` strobe. Some flash devices provide write protection mechanisms, which can be activated during the power-up and reset cycles on the processor. In the absence of

such mechanisms, a pull-up resistor on the AMS0 pin prevents the chip select from floating when the state of the processor is unknown.

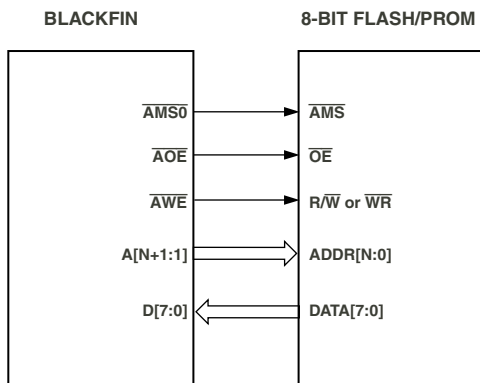


**Figure 34-8:** 8-bit Flash Interconnection

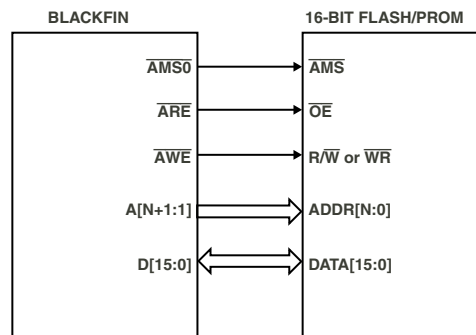


**Figure 34-9:** 16-bit Flash Interconnection

**Table 34-15:** MDMA Codes

| MDMACODE | Word Width | Source Modify | Purpose                                                |
|----------|------------|---------------|--------------------------------------------------------|
| 0x0      |            |               | reserved, protects against legacy 10-byte boot streams |
| 0x1      | 8 bits     | 1             | 8-bit flash on 8-bit SMC/EBIU                          |
| 0x2      | 8 bits     | 2             | 8-bit flash on 16-bit SMC/EBIU                         |
| 0x3      | 8 bits     | 4             | 8-bit flash on 32-bit SMC/EBIU                         |
| 0x4      | 8 bits     | 8             | reserved - not supported                               |
| 0x5      | 8 bits     | 16            | reserved - not supported                               |
| 0x6      | 16 bits    | 2             | 16-bit flash on 16-bit SMC/EBIU                        |
| 0x7      | 16 bits    | 4             | 16-bit flash on 32-bit SMC/EBIU                        |
| 0x8      | 16 bits    | 8             | reserved - not supported                               |
| 0x9      | 16 bits    | 16            | reserved - not supported                               |
| 0xA      | 32 bits    | 4             | 32-bit flash on 32-bit SMC/EBIU                        |
| 0xB      | 32 bits    | 8             | reserved - not supported                               |
| 0xC      | 32 bits    | 16            | reserved - not supported                               |



**Table 34-15:** MDMA Codes (Continued)

| MDMACODE | Word Width | Source Modify | Purpose                  |
|----------|------------|---------------|--------------------------|
| 0xD      | 64 bits    | 8             | reserved - not supported |
| 0xE      | 64 bits    | 16            | reserved - not supported |
| 0xF      | 128 bits   | 16            | reserved - not supported |

To illustrate how the MDMACODE effects reading memory, consider eight contiguous bytes A through H in the following scenarios. In the first, a 16-bit device is used on the 16-bit SMC, stored as BA, DC, FE, HG.

**Table 34-16:** MDMACODE effect on memory read

| MDMACODE | Read by kernel as      |
|----------|------------------------|
| 0x1      | A, B, C, D, E, F, G, H |
| 0x6      | BA, DC, FE, HG         |
| 0xA      | DCBA, HGFE             |

## Padding Memory

To ensure data is read properly for the specific configuration, it may be necessary to pad data when storing it in the device. A few examples follow.

If an 8-bit flash device is connected to the 16-bit SMC only the lower byte is populated and the upper byte needs to be ignored by the kernel. This is ensured by setting MDMACODE = 0x2. To use a 16-bit flash to simulate this, the upper byte is usually zero padded. The 16-bit flash device is programmed with the content of 0A, 0B, 0C, 0D, 0E, 0F, 0G, 0H.

Similarly, setting MDMACODE = 0x3 supports 8-bit flash on 32-bit SMC. This case can be simulated by connecting a 32-bit flash to a 32-bit SMC, or by connecting a 16-bit flash to a 16-bit SMC. In either case, the flash is programmed with three padding bytes per data byte: 000A, 000B, 000C, 000D, 000E, 000F, 000G, 000H.

Finally, setting MDMACODE=0x7 supports 16-bit flash on a 32-bit SMC. This case can also be simulated by connecting a 32-bit flash to a 32-bit SMC, or by connecting a 16-bit flash to a 16-bit SMC. This time 16-bit padding is required per 16-bit data byte: 00BA, 00DC, 00FE, 00HG.

## Auto Detection

An automatic configuration detection is provided that uses the first nibble of the boot stream (MDMACODE field). The boot code requests the initial eight bytes by 32-bit MDMA, inspects the MDMACODE and then adjusts

to 8-bit or 16-bit MDMA operation as needed. During this detection sequence, external memory addresses are fetched sequentially; alternatively a sequential device (such as a FIFO) can be connected.

When the Boot Routine API function is called at run time, the same auto-detection is performed. If auto-detection is not wanted, it can be suppressed by the `BCMD_NOAUTO` switch. See the Run-time API section for details. It is important to understand the `BCMD_NOAUTO` enables the `MDMACODE` to be changed at runtime. If data is arranged on the same flash device in different ways, its perfectly valid to rotate though `MDMACODE` settings 0x01, 0x6 and 0xA, because all three `MDMACODE`s can operate on the same physical arrangement and the same flash image. However, setting `MDMACODE` to 0x2, 0x3 and 0x7 requires a very specific format and are therefore not interchangeable.

## Default Static Memory Controller (SMC) settings

In Memory Boot mode, the boot kernel attempts to boot from the ASYNC Bank 0 Memory Space. Bank 1, Bank 2 and Bank 3 are not initialized. If required, they can be initialized by `initcode`. The SMC registers are set by default as described in the Default SMC Settings table.

**Table 34-17:** Default SMC Settings

| Register Bit       | Value | Note                                |
|--------------------|-------|-------------------------------------|
| SMC_B0CTL_EN       | 1     | enable bank AMS0                    |
| SMC_B0CTL_MODE     | 0     | normal asynchronous mode            |
| SMC_B0CTL_RDYEN    | 1     | enable ARDY                         |
| SMC_B0CTL_RDYPOL   | 1     | ARDY is active high                 |
| SMC_B0CTL_RDYABTEN | 0     | disable ARDY abort counter          |
| SMC_B0CTL_PGSZ     | 0     | 4-byte page size                    |
| SMC_B0CTL_BLK      | 1     | NOR clock = $SCLKA/2$               |
| SMC_B0CTL_BYTPE    | 0     | wrap bursts                         |
| SMC_B0CTL_SELCTRL  | 0     | normal AMS0 strobe                  |
| SMC_B0TIM_WST      | 7     | $7 \times 8\text{ns} = 56\text{ns}$ |
| SMC_B0TIM_WHT      | 7     | $7 \times 8\text{ns} = 56\text{ns}$ |
| SMC_B0TIM_WAT      | 9     | $9 \times 8\text{ns} = 72\text{ns}$ |
| SMC_B0TIM_RST      | 7     | $7 \times 8\text{ns} = 56\text{ns}$ |

**Table 34-17:** Default SMC Settings (Continued)

| Register Bit     | Value | Note                                |
|------------------|-------|-------------------------------------|
| SMC_B0TIM_RHT    | 1     | $1 \times 8\text{ns} = 8\text{ns}$  |
| SMC_B0TIM_RAT    | 8     | $8 \times 8\text{ns} = 64\text{ns}$ |
| SMC_B0ETIM_PREST | 1     | $1 \times 8\text{ns} = 8\text{ns}$  |
| SMC_B0ETIM_PREAT | 3     | $3 \times 8\text{ns} = 24\text{ns}$ |
| SMC_B0ETIM_TT    | 2     | $2 \times 8\text{ns} = 16\text{ns}$ |
| SMC_B0ETIM_IT    | 2     | $2 \times 8\text{ns} = 16\text{ns}$ |
| SMC_B0ETIM_PGWS  | 9     | $9 \times 8\text{ns} = 72\text{ns}$ |

## Run-time API

A Memory Boot can be initiated through the Boot Routine API function at run time. Any start address which is divisible by 4 can be specified. The API allows customizations such as booting from a different SMC bank, and disabling auto-detection.

Memory Boot is selected by specifying the `BCMD_DEVICE_MEMORY | BCMD_MEMORY` in the boot command argument. If `BCMD_DEVENUM_0` is specified, the routine extracts from the start address which SMC bank to configure. If, however, `BCMD_DEVENUM_1` is specified, the `BCMD_MEM_SMCBANK` field selects the SMC bank which is configured.

In either case the routine also initializes pin muxing as required unless the `BCMD_NOCFG` flag is specified, which overrules `BCMD_DEVENUM` and `BCMD_MEM_SMCBANK`. If the `BCMD_NOCFG` flag is specified, it is the programs responsibility to configure SMC and pin muxing as required. This is the only way to boot in burst, page or (the so-called) flash mode. The `BCMD_NOCFG` option is useful if a different application (dx) has been booted earlier and the SMC and pin muxing have already been set up.

The `MDMACODE` configuration detection can be suppressed by the `BCMD_NOAUTO` switch. In this case, the desired configuration can be passed through the `BCMD_MODE` bit field. It follows `MDMACODE` through table convention.

## RSI Master Boot Mode

The RSI master boot mode loads a loader stream through the Removable Storage Interface (RSI). After reset or power on, the boot kernel performs auto-detection of the device to determine its type. Device types supported by auto-detection are:

- Secure Digital (SD), low/high capacity, 1/4-bit
- Multi Media Card (MMC), low/high capacity, 1/4-bit
- MMC plus, low/high capacity, 1/4/8-bit
- Embedded MMC (eMMC) low/high capacity, 1/4/8-bit as aaaaa

The boot kernel optimizes clock settings based on an RSICODE provided in the first nibble of the first block (defined as BCODE code in the block header definition). When booting from RSI, pagemode is used to load data. This provides better support for block based devices (most RSI devices are block based).

Every block loaded in RSI boot mode is loaded implicitly as an indirect, callback block. A callback routine is not required, but may be useful. An indirect callback block causes the boot kernel to first load pages to a temporary buffer where a callback routing may be used to load the data to a file system or perform other modifications. If no callback routine is specified the block is simply loaded. The Boot Routine API allows more advanced customization for booting from RSI. Using this API, auto-detection can be disabled and the device can be specified and configured manually.

## PageMode Customization

Devices attached to the RSI peripheral are typically block based devices. The boot kernel has built-in support for booting from this type of block based device called pagemode. In pagemode data is loaded by fixed block sizes. The default page size used is 512 bytes. This value can be changed using the initcode routine to modify the `dTempByteCount` parameter in the `struct STRUCT_ROM_BOOT_CONFIG` structure. Refer to pagemode and Init Block for details on making these changes.

## Callback Customization

Frequently, devices connected to the RSI have filesystem implemented on them. The motivation for implicitly enabling a callback for every block is to ease implementation on such systems. By default the callback routine is empty and simply returns a value of zero. The standard procedure for loading and registering a callback function should be used early in the stream. See the callback block section for more detailed information on performing this customization.

## RSI Code

When performing auto-detection, the first block of data is loaded using the bus width defined by the RSICODE. The RSI code is defined by the first block in the boot stream and is located in the block header where BCODE is defined in a generic block code, (see Block Code). The boot kernel then reconfigures the RSI peripheral according to the value detected in the RSICODE.

**Table 34-18:** RSI Code

| RSICODE | uwFlag   | ubBusWidth | ubRsiClk | ubRsiClk High Speed | Description                      |
|---------|----------|------------|----------|---------------------|----------------------------------|
| 0       | 0x0000   | 0x00       | 0x3      | N/A                 | 1 bit, regular speed protocol    |
| 1       | 0x0020   | 0x00       | 0x3      | 0x1                 | 1 bit, high speed protocol(1)    |
| 2       | 0x0000   | 0x01       | 0x3      | N/A                 | 4 bit, regular speed protocol    |
| 3       | 0x0020   | 0x01       | 0x3      | 0x1                 | 4 bit, high speed protocol(1)    |
| 4       | 0x0000   | 0x02       | 0x3      | N/A                 | 8 bit, regular speed protocol(2) |
| 5       | 0x0020   | 0x02       | 0x3      | 0x1                 | 8 bit, high speed protocol(1)(2) |
| 6-15    | reserved |            |          |                     |                                  |

## Run-Time API

The RSI Boot mode can also be initiated through the Boot Routine API. The boot routine allows for further customization by modifying values in the `dBootCommand` parameter. This can be useful when it is necessary to bypass auto-detection of the device.

**dBootCommand: dBootCommand Parameter for RSI Master Boot Mode - RW**

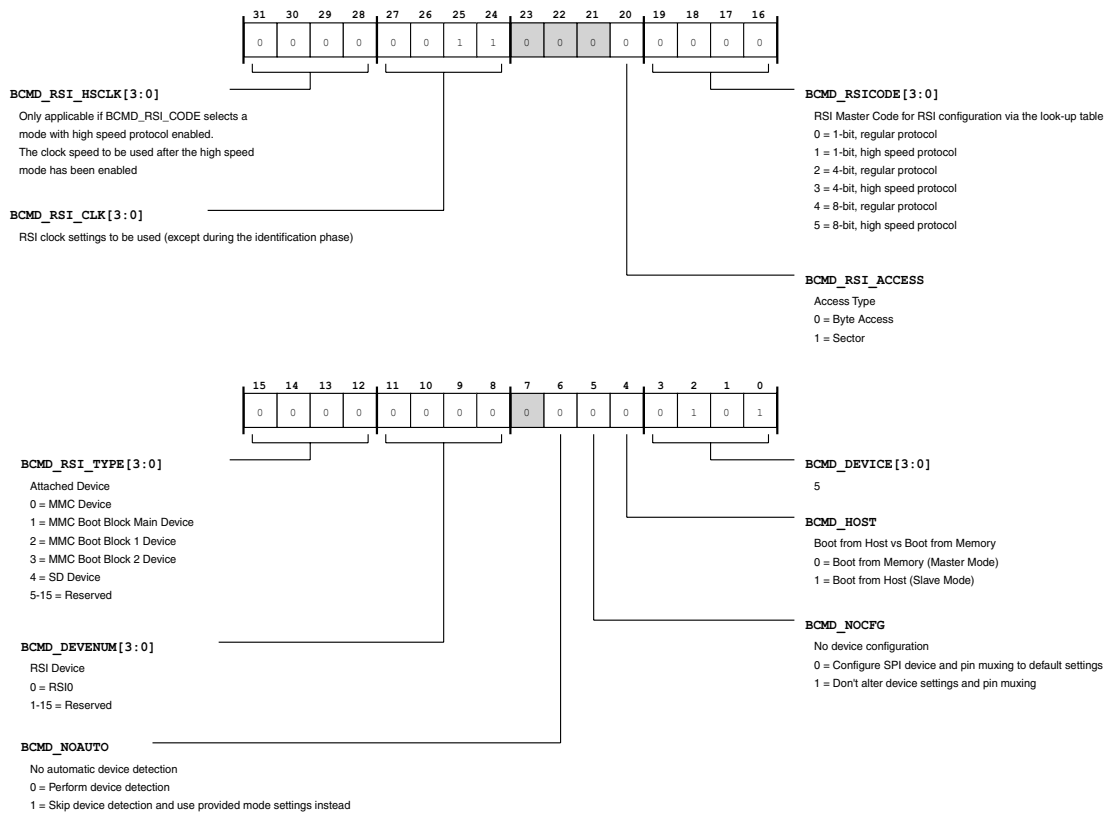


**Figure 34-10: dBootCommand for RSI Boot**

When `BMC_NOAUTO` is set, the RSI clock divider is taken from the `BCM_RSI_CLK` field. The device type of the attached device is taken from the `BCMD_CHANEL` field and the access type is taken from the `ACCESS` field. The `RSICODE` field is still applicable and processed in order to set the bus width and speed protocol to use.

**NOTE:** The device identification phase is not executed. Therefore the attached device must be in the transfer state before executing the API call.

In order to communicate with an attached device that is in the transfer state, the RCA of the device is required. For example, `CMD13` reads the card's status in the event the switch command was executed to change the bus width or timings of the device. The card's RCA is therefore required to be written to the `uwRca` field of the RSI structure via the use of the hook function that can be passed to the boot kernel via the API. Failure to do so may lead to a boot failure

For MMC devices that support boot blocks, the contents of byte 179 of the Extended CSD register should be written to the `ubBootPartByte` of the RSI structure. This should be preformed via the hook function. In particular bits [7:3] should be set accordingly. Bits [2:0] may be modified by the boot code depending on the device passed via the `BCMD_RSI_TYPE` field of `dBootCommand`.

The boot kernel uses a byte write operation to write byte 179 of the Extended CSD register which impacts the non-volatile bits of the byte. When `BCMD_NOAUTO` is set, the Extended CSD register is not read which speeds up the process. Set the non-volatile bits accordingly. Programs may do this by reading the Extended CSD register at some point in the application and passing in byte 179 accordingly.

## Notes on eMMC

The boot kernel includes support to handle boot partitions on eMMC devices. The extended CSD register is a register on the eMMC device with a byte containing boot partition information. The boot kernel uses this byte to activate the required partition accordingly to then get the boot stream. While the boot partition may be small, it may be sufficient to store something like UBOOT in loader stream format. UBOOT can be loaded from this special boot partition and then it can continue to boot having knowledge of the main file system are of the eMMC.

## SPI Master Boot Mode

Describes booting from the Serial Peripheral Interface (SPI).

This SPI Master Boot mode boots from SPI memory connected to the SPI0 interface. 8-bit, 16-bit, 24-bit and 32-bit address words are supported. Standard SPI memory is read using either the standard 0x03 SPI read command or the 0x0B SPI fast read command.

For booting, the SPI memory is connected as shown in [SPI Memory Connections](#)

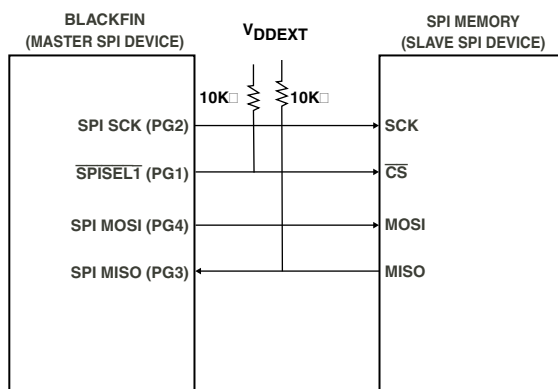


**Figure 34-11:** SPI Memory Connections

The pull-up resistor on the MISO line is required for automatic device detection. The pull-up resistor on the SEL1 line ensures that the memory is in a known state when the GPIO is in a high-impedance mode such as during reset. A pull-down resistor on the SCK line displays cleaner oscilloscope plots during debugging.

For SPI master boot the `SPE`, `MSTR` and `SZ` bits are set in the `SPI0_CTL` register. The `TIMOD=2` bits enable the receive DMA mode. Clearing both the `CPOL` and `CPHA` bits results in SPI mode 0. The boot kernel does not allow SPI0 hardware to control the `SEL1` pin. Instead, this pin is toggled in GPIO mode by software.

Initcodes are allowed to manipulate the `uwSsel` variable in the [STRUCT\\_ROM\\_BOOT\\_CONFIG](#) structure to extend the boot mechanism to a second SPI memory connected to another GPIO pin.

## SPI Device Detection Routine

Since `BMODE = 011` supports booting from various SPI memories, the boot kernel automatically detects what type of memory is connected. To determine whether the SPI memory device requires an 8, 16, 24 or 32-bit addressing scheme, the boot kernel performs a device detection sequence prior to booting. The MISO signal requires a pull-up resistor, since the routine relies on the fact that memories do not drive their data outputs unless the right number of address bytes are received.

Initially, the boot kernel transmits a read command (either `0x03` or `0x0B`) on the MOSI line, which is immediately followed by two zero bytes. Once the transmission is finished, the boot kernel interrogates the data received on the MISO line. If it does not equal `0xFF`, the valid byte (`0x1-0xE` in the lower nibble) tells the boot code whether the memory device requires 8, 16, 24, 32 address bits. This is referred to as the `SPIMCODE`. The boot kernel has the following settings according to the [SPIMCODE Descriptions](#) table.

If the received value equals `0xFF`, it is assumed that the memory device has not driven its data output and that the `0xFF` value is due to the pull-up resistor. Thus, another zero byte is transmitted and the received data is tested again.

If the value still equals `0xFF`, device detection continues. Device detection aborts immediately if a byte different than `0xFF` is received. The boot kernel continues with normal boot operation and it reissues a read command to re read from address 0. The first block header is loaded by two read sequences, further block headers and block payload fields are loaded by separate read sequences.

Figure [SPI Device Detection Principle](#) illustrates how individual devices behave.

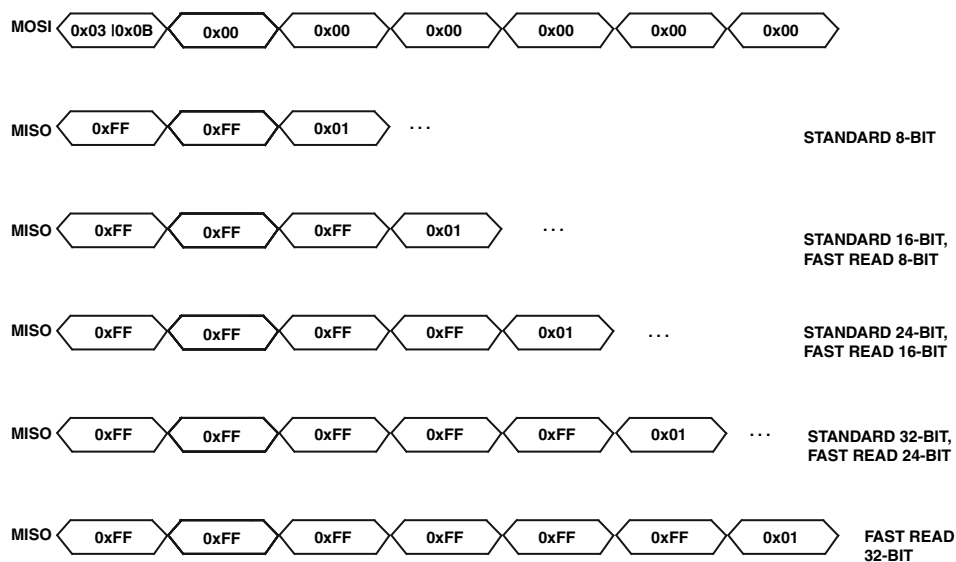


**Figure 34-12:** SPI Device Detection Principle



**Table 34-19:** SPIMCODE Descriptions

| SPIMCODE | Mode                   | Comm<br>and | Dummy<br>Bytes | Data<br>Lines | Address<br>Lines | SPI Clock | Purpose                                  |
|----------|------------------------|-------------|----------------|---------------|------------------|-----------|------------------------------------------|
| 0x0      |                        |             |                |               |                  |           | unused                                   |
| 0x1      | STANDARD               | 0x03        | 0              | 1             | 1                | SCLK0/32  | legacy single-bit SPI mode               |
| 0x2      | STANDARD               | 0x03        | 0              | 1             | 1                | SCLK0/5   | legacy single-bit SPI mode               |
| 0x3      | STANDARD               | 0x0B        | 1              | 1             | 1                | SCLK0/2   | single bit with dummy address byte       |
| 0x4      | FAST_MODE              | 0x0B        | 1              | 1             | 1                | SCLK0/2   | single bit with dummy address bytes      |
| 0x5      | FAST_MODE              | 0x03        | 0              | 1             | 1                | SCLK0/3   | Atmel mode                               |
| 0x6      | FAST_MODE              | 0x0B        | 1              | 1             | 1                | SCLK0/1   | Atmel mode with dummy address byte       |
| 0x7      | FAST_MODE (RAPIDS)     | 0x1B        | 2              | 1             | 1                | SCLK0/1   | Atmel mode with 2 dummy address bytes    |
| 0x8      | DOR FAST_MODE          | 0x3B        | 1              | 2             | 1                | SCLK0/2   | dual bit data, single bit address        |
| 0x9      | DIOR FAST_MODE         | 0xBB        | 1              | 2             | 2                | SCLK0/2   | dual bit data, dual bit address          |
| 0xA      | QOR FAST_MODE (Atmel)  | 0x6B        | 1              | 4             | 1                | SCLK0/2   | quad bit data, single-bit address        |
| 0xB      | QIOR FAST_MODE (Atmel) | 0xEB        | 3              | 4             | 4                | SCLK0/2   | quad bit data, quad-bit address (AT25DQ) |

**Table 34-19:** SPIMCODE Descriptions (Continued)

| SPIMCODE | Mode              | Comm<br>and | Dummy<br>Bytes | Data<br>Lines | Address<br>Lines | SPI Clock | Purpose                              |
|----------|-------------------|-------------|----------------|---------------|------------------|-----------|--------------------------------------|
| 0xC      | QOR<br>FAST_MODE  | 0x6B        | 1              | 4             | 1                | SCLK0/2   | quad bit data,<br>single-bit address |
| 0xD      | QIOR<br>FAST_MODE | 0xEB        | 3              | 4             | 4                | SCLK0/2   | quad bit data, quad<br>bit address   |
| 0xE      | reserved          |             |                |               |                  |           |                                      |
| 0xF      |                   |             |                |               |                  |           | unused                               |

## Run-time API

The Boot Routine API function can be used to initiate a boot through the SPI device. Using the API, further customizations can be made such as selecting a device other than SPI0, or disabling automated device initialization and pin muxing. Refer to the specific flags in the [dBootCommand for RSI Boot](#) parameter.

**dBootCommand: dBootCommand Parameter for SPI Master Boot Mode - RW**

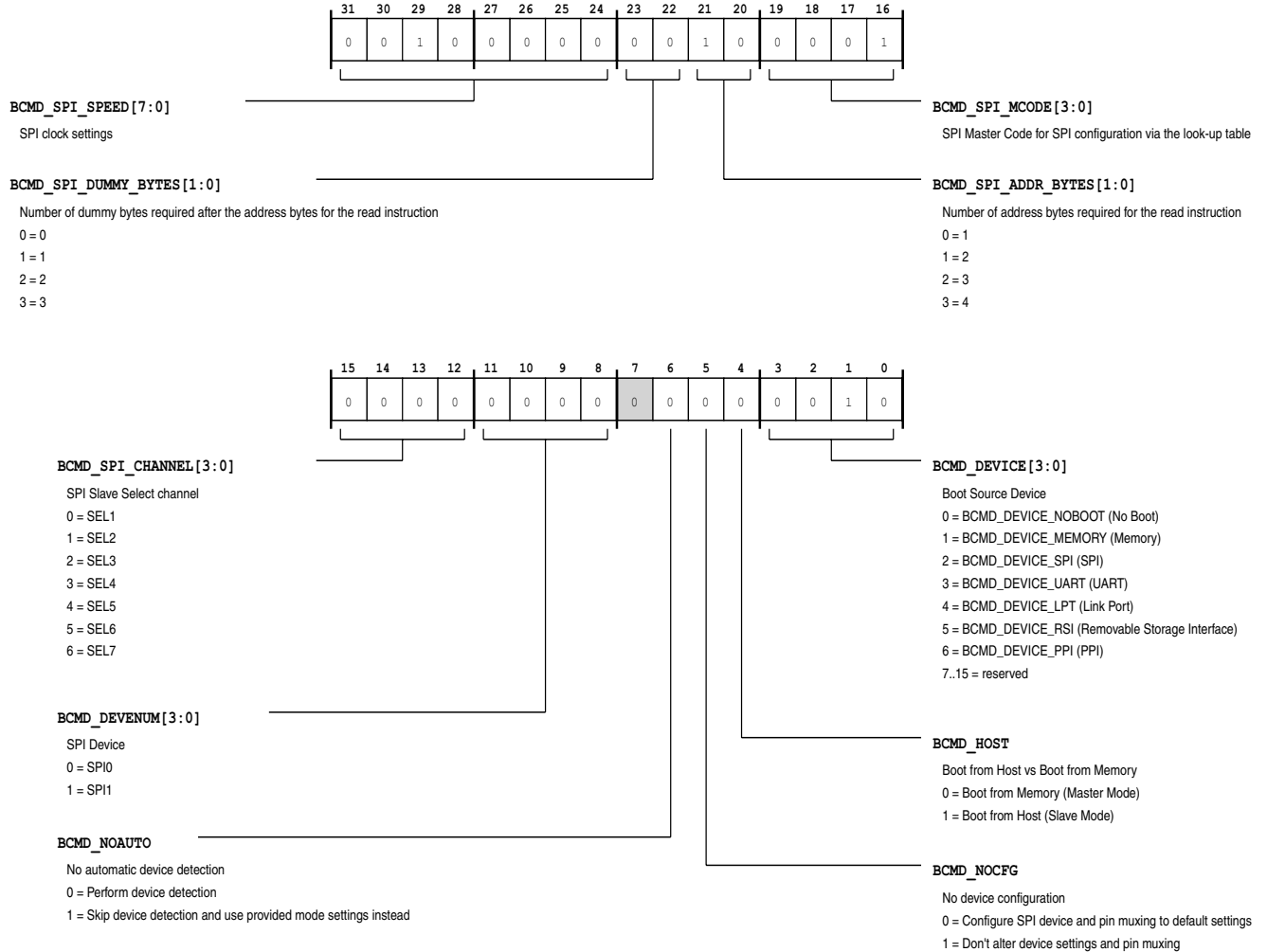


**Figure 34-13: dBootCommand for RSI Boot**

## SPI Slave Boot Mode

Describes booting from the Serial Peripheral Interface (SPI) with the processor as a slave.

When using SPI slave mode boot, the processor consumes boot data from an external SPI host device. This mode supports single, dual, and quad-bit modes. The boot kernel always starts in single bit mode and can be changed using the appropriate command. The hardware configuration for the modes is shown in the following figures. As in all slave boot modes, the host device controls the Blackfin processor's RESET input.

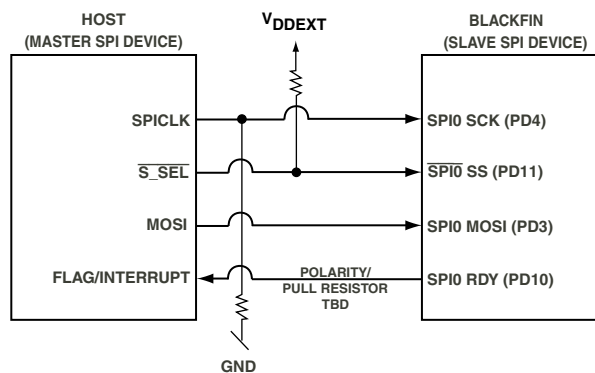


**Figure 34-14:** Connection Between Host (SPI Master) and Processor (SPI Slave)

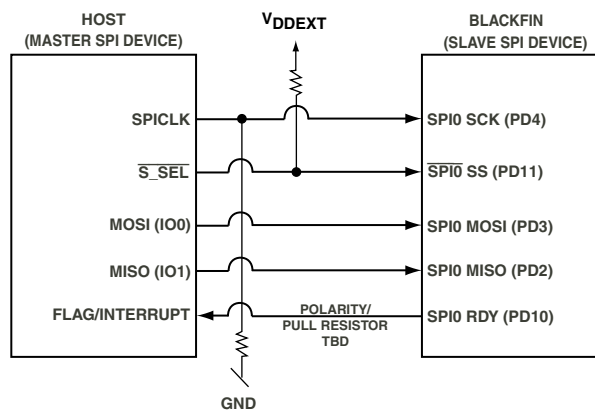


**Figure 34-15:** Connection Between Host (SPI Master) and Processor (SPI Slave) DIOM

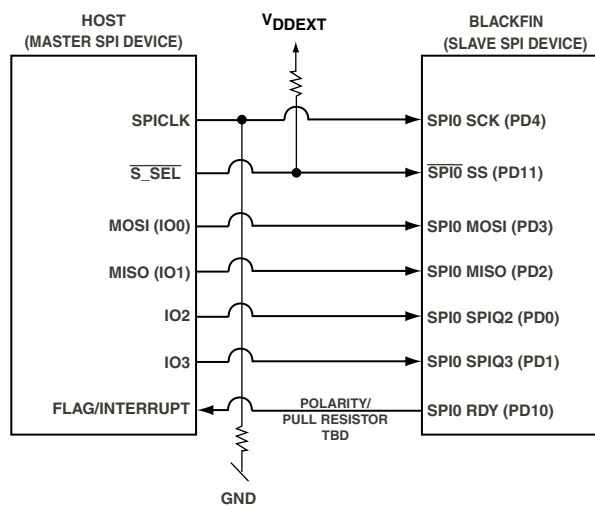


**Figure 34-16:** Connection Between Host (SPI Master) and Processor (SPI Slave) QSPI

The host drives the SPI clock and is responsible for timing. The host must provide an active-low chip select signal that connects to the SPI0 SS input of the Blackfin processor. It can toggle with each byte transferred or remain low during the entire procedure. 8-bit data is expected and 16-bit mode is not supported.

In SPI slave boot mode, the boot kernel sets the CPHA bit and clears the CPOL bit in the SPI\_CTL register. Therefore the MISO pin is latched on the falling edge of the MOSI pin. For details see "SPI Compatible Port Controllers" in the Processor Hardware Reference.

The SPI slave processor detects the correct bit mode from the host SPI device by reading the first byte sent, defined as the SPICMD. The following table describes the available codes. If the host starts in dual or quad-bit mode, additional bytes need to be sent to transmit the correct code.

**Table 34-20:** SPICMD Descriptions

| SPICMD                                | Description             |
|---------------------------------------|-------------------------|
| Starting in Single bit Mode           |                         |
| 0x3                                   | keep single-bit mode    |
| 0x7                                   | switch to dual-bit mode |
| 0xB                                   | switch to quad-bit mode |
| If host device starts in DIOM or QSPI |                         |
| 0xAA,0xBF                             | switch to dual-bit mode |
| 0xEE,0xEE,0xFE,0xFF                   | switch to quad-bit mode |

In SPI slave boot mode, SPI\_RDY functionality is critical. The SPI\_RDY output is used for back pressure and requires a pull-up resistor. When high, the resistor shown in Figure programs SPI\_RDY to hold off the host. SPI\_RDY holds the host off while the Blackfin processor is in reset or executing the preboot. Once SPI\_RDY turns inactive, the host can send boot data. The SPI module does not provide very large receive FIFOs, so the host must test the HWAIT signal for every byte. Figure [SPI Program Flow on the Host Side](#) illustrates the required program flow on the host side.

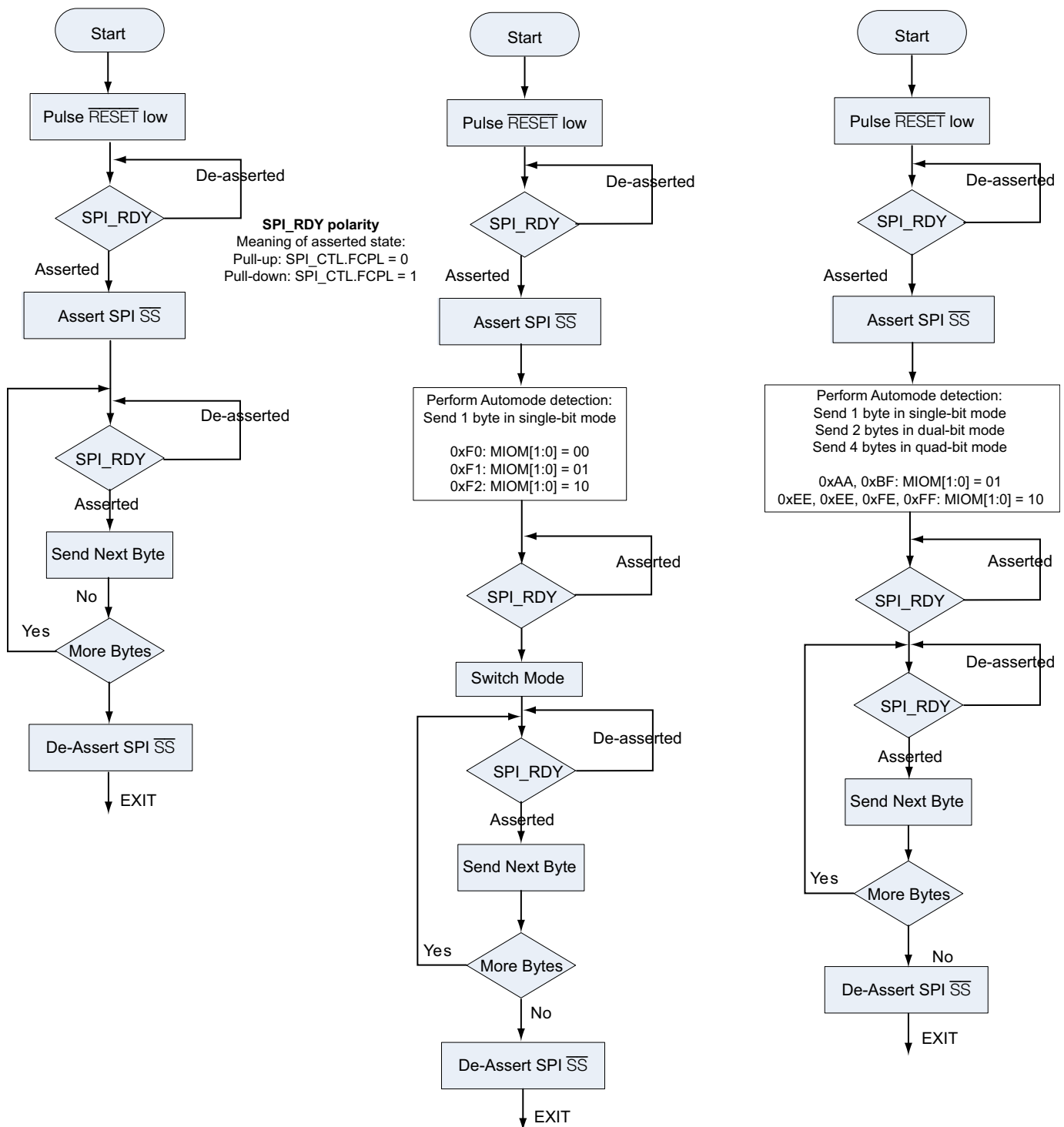


**Figure 34-17:** SPI Program Flow on the Host Side

Run-Time API

The SPI Slave Boot mode can be called through the Boot Routine API function at run time. Initiating a boot through the run-time API allows for additional customization such as disabling automatic device configuration or specifying a different SPI device other than SPI0.

When BFROM\_BCMD\_NOCFG flag is specified, it is necessary to program pin muxing and other SPI configuration as required, while keeping the SPI\_CTL.EN bit cleared.

The automode detection can be suppressed by the BFROM\_BCMD\_NOAUTO switch. In that case, the desired configuration must be passed through the BFROM\_BCMD\_SPI\_CODE bit field, even if the BFROM\_BCMD\_NOCFG flag is set.

Figure *dBootCommand* describes the fields possible for customization through the dBootCmd parameter.

dBootCommand: dBootCommand Parameter for SPI Slave Boot Mode - RW

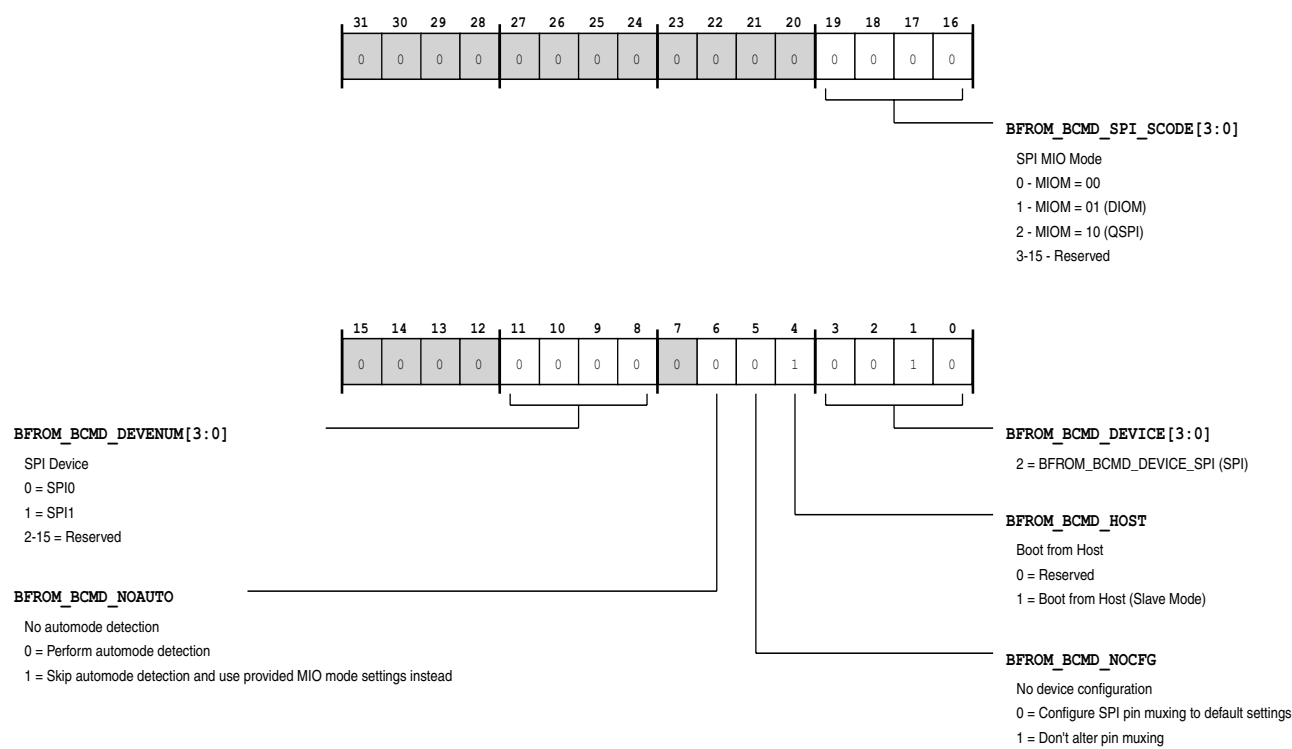


Figure 34-18: dBootCommand

Link Port Slave Boot Mode

Describes booting from the Link Port with the processor as a slave

Link port boot is a slave boot mode in which the processor receives boot data from an external link port master through link port 0. The link port is configured for receive mode and all transfers from the link port to memory are performed under the control of DMA. The maximum supported operating frequency of the Link Port is 66MHz for which the master boot source is responsible for deriving the clock frequency. The link port receiver operates at an asynchronous frequency up to the maximum supported operating frequency.

The link port protocol supports a means of generating link port transmit and receive service requests. The transmit service request is generated on the processor to transmit the data by the receiver driving the `LACKx` signal high when the transmitter is disabled. The receive service request is generated on a receiver when it is disabled. This is initiated by the transmitter driving the `LCLKx` signal high.

As transmitter and receivers may be enabled at different times external pull-down resistors are required on both the `LCLKx` and `LACKx` signals in order to eliminate any false service request assertions.

The link port slave boot mode initialization phase waits for the receive service request before passing control back to the main kernel. Once this initial receive service request has been detected the receiving link port is enabled and the boot process completes. At no point prior to boot completion is the receiving link port disabled again. Once the link port is enabled all transfers are controlled by the receive DMA channel. The Load function for the link port Receive Boot mode may therefore simply point to the peripheral DMA routine of the main kernel in a similar way to the SPI slave boot mode.

## **Run-time API**

The Boot Routine API function can be used to load a boot stream through the link port peripheral. Using this API the specific LP device may also be chosen, otherwise only link port0 is assumed. See the kernel API section for specifics on how to use this function.



### dBootCommand: dBootCommand Parameter for Linkport Boot Mode - RW

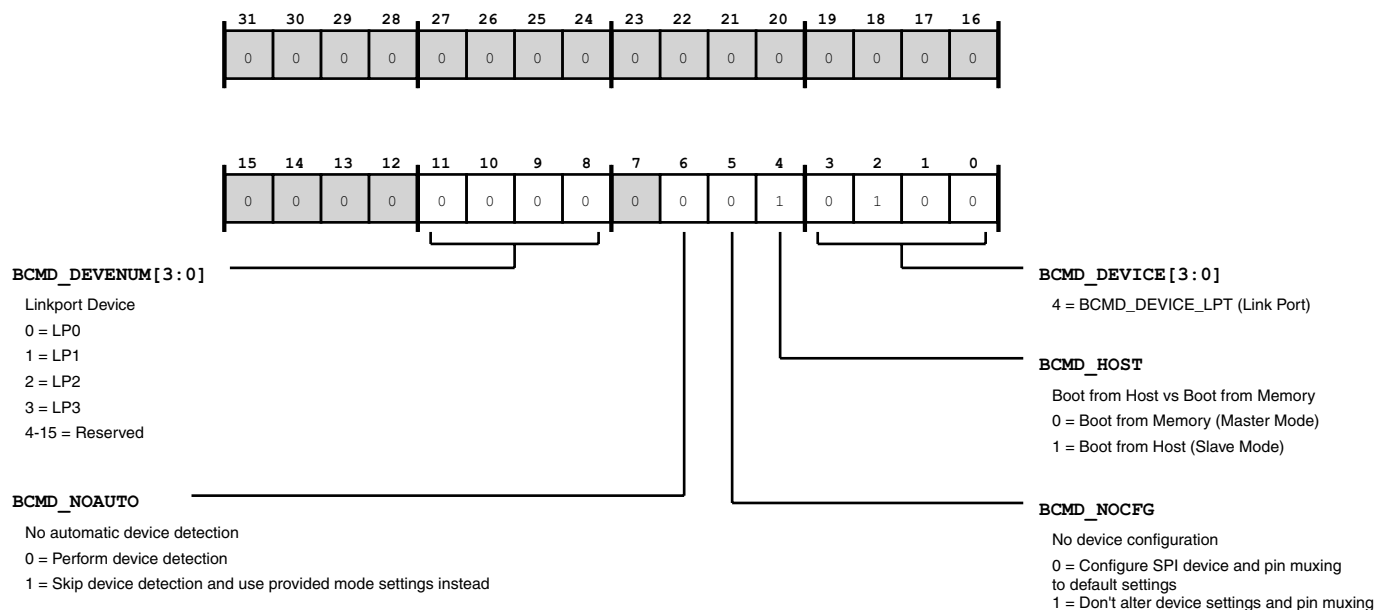


**Figure 34-19:** dBootCommand for Link Port Boot

## UART Slave Boot Mode

When using UART slave mode boot, the processor receives boot data from a UART host device connected to the UART interface. The device connected to UART0 is initially detected using an autobaud detection sequence. After finishing the UART slave boot process, all control and status registers of the used resources are restored.

Further customization, such as disabling autobaud detection, and changing the device, can be done using the Boot Routine API.

During the boot operation, the host device usually relies on the **RTS** output of the UART device. At boot time the processor does not evaluate RTS signals driven by host. Since the RTS is in a high impedance state when the processor is in reset, or while executing a preboot, an external pull-up resistor to **VDDEXT** is recommended. The Connection Between Host and Processor figure shows the interconnection required for booting. The figure does not show physical line drivers and level shifters that are typically required to meet the individual UART-compatible standards.

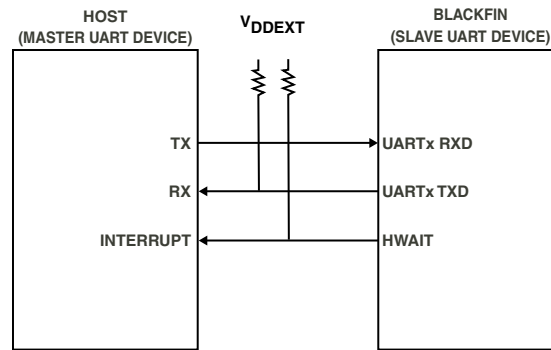


**Figure 34-20:** Connection Between Host and Processor

When the UART is enabled, the RTS goes immediately low, encouraging the host to send the first boot stream data as shown in the figure. In the case of half-duplex UART connections, this must be avoided. The host should wait until it has received the four bytes from the slave processor, before sending any data.

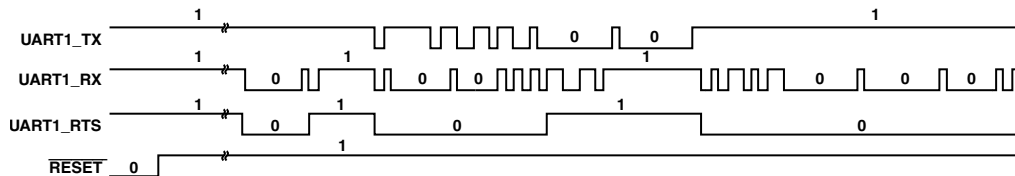


**Figure 34-21:** Host relying on RTS

When the boot kernel is processing fill or initcode blocks, it might require additional processing time and needs to hold the host off from sending more data. This is signaled using the RTS output.

The figure above shows RTS timing in case an extended initcode routine executes. Since code execution is distracting from the data loading, the host device has to be prevented from sending more data. The timing of the RTS depends on the state of the RFRT bit in the UART Control register (UART\_CTL). This bit is cleared in case of the UART Slave Boot mode and RTS is deasserted when the UART receive FIFO contains 4 or more data words and another start bit is detected.

## Autobaud Detection

The kernel supports autobaud detection using the '@' character as data. The host is expected to have its clock set to rate supported in the UART

To determine the bit rate when performing the autobaud:

1. the boot kernel expects an '@' character (0x40, eight bits data, one start bit, one stop bit, no parity bit) on the UART RXD input.
2. The EDB0 and UART\_CLK register is cleared.
3. The boot kernel acknowledges, and the host then downloads the boot stream. The acknowledgment consists of four bytes: 0xBF, UART\_CLK[15:8], UART\_CLK[7:0], 0x00.

4. The host is requested to not send further bytes until it has received the complete acknowledge string.
5. Once the 0x00 byte has been received, the host can send the entire boot stream.

The host should know the total byte count of the boot stream, but it is not required to have any knowledge about the content of the boot stream.

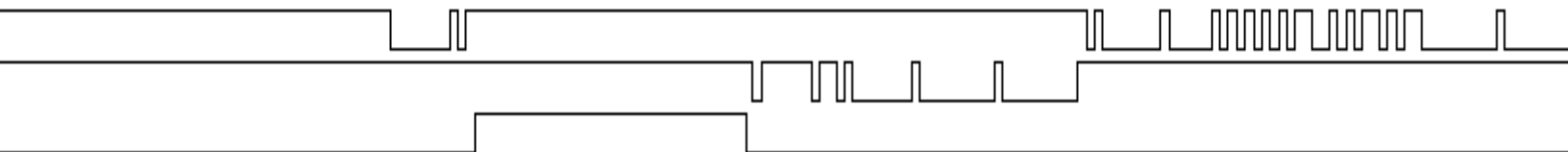


**Figure 34-22:** UART Autobaud Detection Waveform

The UART Autobaud Detection Waveform figure provides timing information for UART booting. After the bit rate is known, the UART is enabled and the kernel transmits four bytes.

## Run-time API

The UART Slave Boot mode can be called through the Boot Routine API function at run time. The run-time API allows for additional customization. both autobaud detection and device configuration can be disabled, and a device other than the default, UART0, may be specified.

If `BFROM_BCMD_NOCFG` flag is specified, it is the programs responsibility to configure pin muxing as required.

Autobaud detection can be suppressed using the `BFROM_BCMD_NOAUTO` switch. In this case, the desired configuration can be passed through the `BFROM_BCMD_UART_CLK` bit field. If the `BFROM_BCMD_UART_CLK` bit field is zero, `UART_CLK` finally is evaluated. If the reset value is detected, the default error routine of the boot kernel is called and the booting process is aborted. Otherwise the value in `UART_CLK` remains untouched.

The `dBootCommand` figure shows each of the available fields in the `dBootCommand` parameter in the Boot Routine API function.

dBootCommand: dBootCommand Parameter for UART Slave Boot Mode - RW

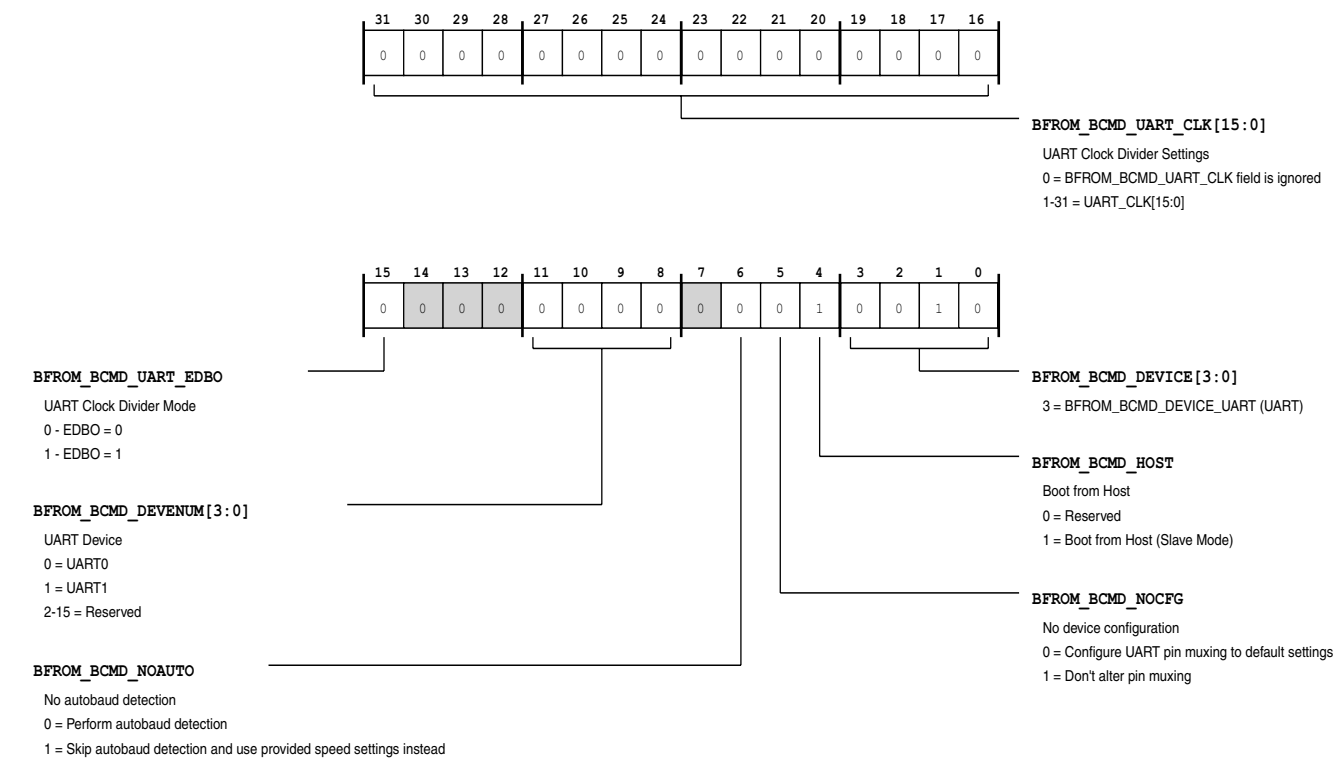


Figure 34-23: dBootCommand

# Boot Programming Model

This section describes the programming model for booting the processor. The programming model includes booting functions, API calls, and data structures.

## Load Functions

All boot modes are processed by a common boot kernel algorithm. The load function point is exposed by the kernel to allow more complete customization of the booting process. The major customization is done by a subroutine that must be registered to the pLoadFunction pointer in the [STRUCT\\_ROM\\_BOOT\\_CONFIG](#) structure. Its simple prototype is as follows.

```
void LoadFunction (STRUCT\_ROM\_BOOT\_CONFIG* pBootStruct);
```

For some scenarios some of the flags in the dFlags word of the [STRUCT\\_ROM\\_BOOT\\_CONFIG](#) structure, such as BFLAG\_PERIPHERAL and BFLAG\_SLAVE, slightly modify the boot kernel algorithm.

The boot ROM contains several load functions. One performs a memory DMA for memory boot, another performs a peripheral DMA. The first is reused for fill operations and indirect booting as well.

In second-stage boot schemes, programs can create customized load functions or reuse the originals and modify the `pDmaControlRegister`, `pControlRegister` and `nControlValue` values in the `STRUCT_ROM_BOOT_CONFIG` structure. The `pDmaControlRegister` points to the `DMAx_CONFIG` or `MDMA_Dx_CONFIG` registers. When `BFLAG_SLAVE` flag is not set, the `pControlRegister` and `dControlValue` variables instruct the peripheral DMA routine to write the control value to the control register every time the DMA is started.

Custom load functions must meet the following requirements.

- Protect against `dByteCount` values of zero
- Multiple DMA work units are required if the `dByteCount` value is greater than 65536
- The `pSource` and `pDestination` pointers must be properly updated

In slave boot modes, the boot kernel uses the address of the `dArgument` field in the `pHeader` block as the destination for the required dummy DMA's when payload data is consumed from `BFLAG_IGNORE` blocks. If the load function requires access to the block's `ARGUMENT` word, it should be read early in the function.

## Page Mode

For the benefit of page oriented boot source devices such as RSI, the boot kernel provides support for page operations. Page mode optimizes memory reads for block organized devices by always reading a page, rather than reading data on demand. The same temporary buffer used by the indirect blocks is used in page mode. The size of the buffer is defined by the `dTempByteCount` variable in the `STRUCT_ROM_BOOT_CONFIG` structure. The page size of the physical source device is defined by the `dPageByteCount` variable.

The `pTempSource` variable points to the source address of the data that is currently in the temp buffer. This variable is an internal variable of the boot kernel. However, at any time programs can set this variable to -1 to force the kernel to re fetch source data into the temp buffer. The following items also pertain to Page-mode.

- `dPageByteCount` must be a power-of-2 value (default is 4)
- `dTempByteCount` must be the same as or a multiple of `dPageByteCount` (default is 512)
- `pTempBuffer` does not have special alignment requirements. However, alignment of 32 may speed up DMA operations.
- In page mode, both block payload data and block headers are loaded through the same mechanism.

## Changing Settings at Run Time

Programs can change the `pTempBuffer`, `dTempByteCount`, and `dPageByteCount` variables at any time, even within `initcodes` or `callbacks`. Whenever the settings change the kernel continues to operate on the

old settings until the content of the former temporary buffer has been entirely processed. The new settings only become active for the next load operation. The kernel can be forced to immediately switch to the new settings by setting `pTempSource` to a value of -1. Note that doing so requires the kernel to re fetch data that had been loaded earlier. Under normal conditions `pTempSource` should not be altered.

The [Initcode Routine](#) example illustrates on how page mode can be activated in any boot mode. Think of SPI master mode that has significant overhead when fetching data in little chunks, and for block headers or small boot blocks.

procedure initcode (BOOT\_CONFIG config)

```
# The config input is the boot config datastructure
# change PLL and initialize DDR controller first */
initPLL()
# enable page mode operation
set config.dFlags.BITM_ROM_BFLAG_PAGEMODE to true;

# set to any unused DDR address,
# make sure it is not overwritten by the boot stream
# ideally pointing to an uninitialized section, such as the stack or heap

    set config.pTempBuffer to 0xABCDABCD;
#update the temp byte count
# pSource points to the current source address
# pNextDxe points to the first address after the DXE
set config.dTempByteCount to config.pNextDxe - config.pSource
```

**Figure 34-24:** Initcode Routine Pseudo-Code

## CRC32 Protection

This section describes the CRC32 Protection provisions

The boot kernel provides mechanisms to allow each block to be verified using a 32-bit CRC. To enable this feature, use the kernel initcode routine, `rom_Crc32Initcode`. An init block that provides this function can be created as the target address, and the ARGUMENT field containing the CRC32 checksum polynomial. Once this function is called, CRC verification is enabled for all blocks except forward, ignore, and first blocks. See the Boot Kernel API documentation for the specifics of this Initcode function.

## Error Handler

This section describes how to customize the error handler

While the default handler puts the processor into idle mode, an initcode routine can register a customized error function by overwriting the error function pointer to create a customized error handler. The expected prototype is:

```
void ErrorFunction(STRUCT_ROM_BOOT_CONFIG* pBootStruct, void *pFailingAddress);
```

Use an Initcode Routine (see *Block Types*) to write the entry address of the error routine to the `pErrorFunction` pointer in the `STRUCT_ROM_BOOT_CONFIG` structure. The error handler has access to the entire boot structure and receives the instruction address that triggered the error.

The default Error Routine performs the following actions in the following order:

1. Issues `EMUEXCPT` instruction
2. Issues `INTR_SOFT3` software interrupt to SEC
3. Toggles `FAULTb` pin
4. Issues `0xDEAD4400` value to former `CHIPID` register (`0xFFC00014`) to halt the simchip simulator
5. Issues an `IDLE` instruction
6. While (CC) loop;
7. Returns (RTS)

## Fault Management

Unless the `RCU_BCODE_NOFAULTS` or `RCU_BCODE_HALT` registers are set, the main routine enables the SEC and configures the following interrupts as faults early in the process:

```
CGU0_ERR, SEC0_ERR, WDOG0_EXP, DMAC_ERR, CRC0_ERR, SOFT3, C0_DBL_FAULT,  
C0_HW_ERR:= SCTL_SEN | SCTL_FEN;
```

For dual-core devices like the ADSP-BF60x family the Core 1 faults are also activated:

```
WDOG1_EXP, C0_BDL_FAULT, C0_HW_ERR:= SCTL_SEN | SCTL_FEN;
```

After memory initialization, the memory protection channels are enabled as faults:

```
L2CTL0_ECC_ERR, INTR_C0_L1_PARITY_ERR and in the dual-core case  
INTR_C1_L1_PARITY_ERR
```

Events on any of these signals during the boot process will assert the `FAULT` and `FAULTb` output pins after a delay. A delay of 256 cycles is used to give the boot code opportunity to self detect the error situation and to execute an `emuexcpt` instruction to alert the emulator if connected.

```
SEC0_FDLY and SEC_FSRDLY are set to 0x100 value. SEC0_FCTL:= FIEN | FOEN | EN;
```

Note that no interrupt is forwarded to either core.

If the boot code detects an error by software, its default error handler sets the INTR\_SOFT3 software interrupt after having executed an EMUEXCPT and before executing an IDLE.

The boot code does not disable the SEC and fault settings on exit so that a safety hole is not introduced when transitioning to user application.

## Callable API Overview

Describes the kernel API available at run-time

The boot code stored in ROM exposes several functions that can be used during run-time or within init-code or callback routines. This section describes the available functions and how they can be used. All functions meet the C runtime calling conventions described in the C/C++ Compiler and Libraries Manual. C prototypes are provided through `cdef_rom.h` header file, and addresses are provided by `thedef_rom.h` header file.

## System Control

The Syscontrol API provides a consistent reliable method to modify the PLL settings of the processor. It is recommended that this is the only method used to modify the PLL settings.

```
uint32_t rom_SysControl(uint32_t dActionFlags, STRUCT_ROM_SYSCTRL *pSysCtrlSettings,  
void *reserved);
```

### Functional Description

|                    |                                                                                                                           |
|--------------------|---------------------------------------------------------------------------------------------------------------------------|
| PP Define          | FUNC_ROM_SYSCONTROL                                                                                                       |
| Prototype          | <code>uint8_t rom_SysControl(uint32_t dActionFlags,<br/>ADI_SYSCTRL_VALUES *pSysCtrlSettings, void<br/>*reserved);</code> |
| Arguments          | R0: Pointer to Syscontrol flags<br>R1: Pointer to Syscontrol structure<br>R2: reserved word                               |
| Return Value       | R0: Syscontrol error flags                                                                                                |
| Stack Requirements | valid stack (SP, FP dividable by 4) required.                                                                             |

The Syscontrol API provide a normalized method to control setting in the CGU and DDR. The following three types of operations are provided:

- CGU Configuration
- DDR initialization from wakeup event



- CGU and DDR context save and restore operations from the DPM\_RESTOREn registers in order to support entry to hibernate and wakeup from hibernate functionality

Syscontrol does not provide support for:

- Initial configuration of the DDR controller after a power up or reset sequence. The sequence of operations performed for the DDR. Configuration functionality is only compliant with the procedure required in order to restore the DDR configuration in the event of a hibernate/wakeup sequence while the DDR is in the self refresh state.
- Setting the DDR into self refresh mode. The DDR must be set to self refresh mode prior to calling syscontrol in order to save the DDR configuration in preparation of entering hibernate.

Syscontrol has not been designed in such a way that it performs all required operations in a single call. The syscontrol functionality can be broken down into two main categories:

- CGU configuration and limited DDR configuration
- CGU and DDR context save and restore from the DPM\_RESTOREn registers

Syscontrol provides a means of configuring the CGU per user requirements. The CGU\_CTL and CGU\_DIV registers are used to manipulate the various clock frequencies of the system. The [CGU Configuration Capacity](#) table defines the order of the sequences when writing to these two registers.

**Table 34-21:** CGU Configuration Capacity

| CGU_CTL Frequency change | CGU_DIV Frequency change | CGU_DIV_UPDT | CGU_DIV_ALGN | Register Written     |
|--------------------------|--------------------------|--------------|--------------|----------------------|
| No                       | No                       | NA           | NA           | None                 |
| No                       | Yes                      | 1            | 1            | CGU_DIV              |
| Yes                      | No                       | NA           | NA           | CGU_CTL              |
| Yes                      | Yes                      | 0            | 0            | CGU_DIV then CGU_CTL |

*When the CGU\_STAT register indicates that the PLL is disabled then CGU\_DIV\_ALGN is never set.*

CGU and DDR context save and restore operations from the DPM\_RESTOREn registers are enabled with the setting of ROM\_SYSCtrl\_WUA\_EN in the [dActionFlags](#) parameter. When set any CGU or DDR configuration operations are ignored. Therefore any attempt to reconfigure the PLL while this bit is set will not be performed.

The ROM code calls syscontrol itself only when the device is waking up from hibernate. Nevertheless, syscontrol is an important primitive of the boot concept as programs are expected to call syscontrol from within their initcode to speed-up the boot process.

dActionFlags: Syscontrol Action Flags (R/W) - RW

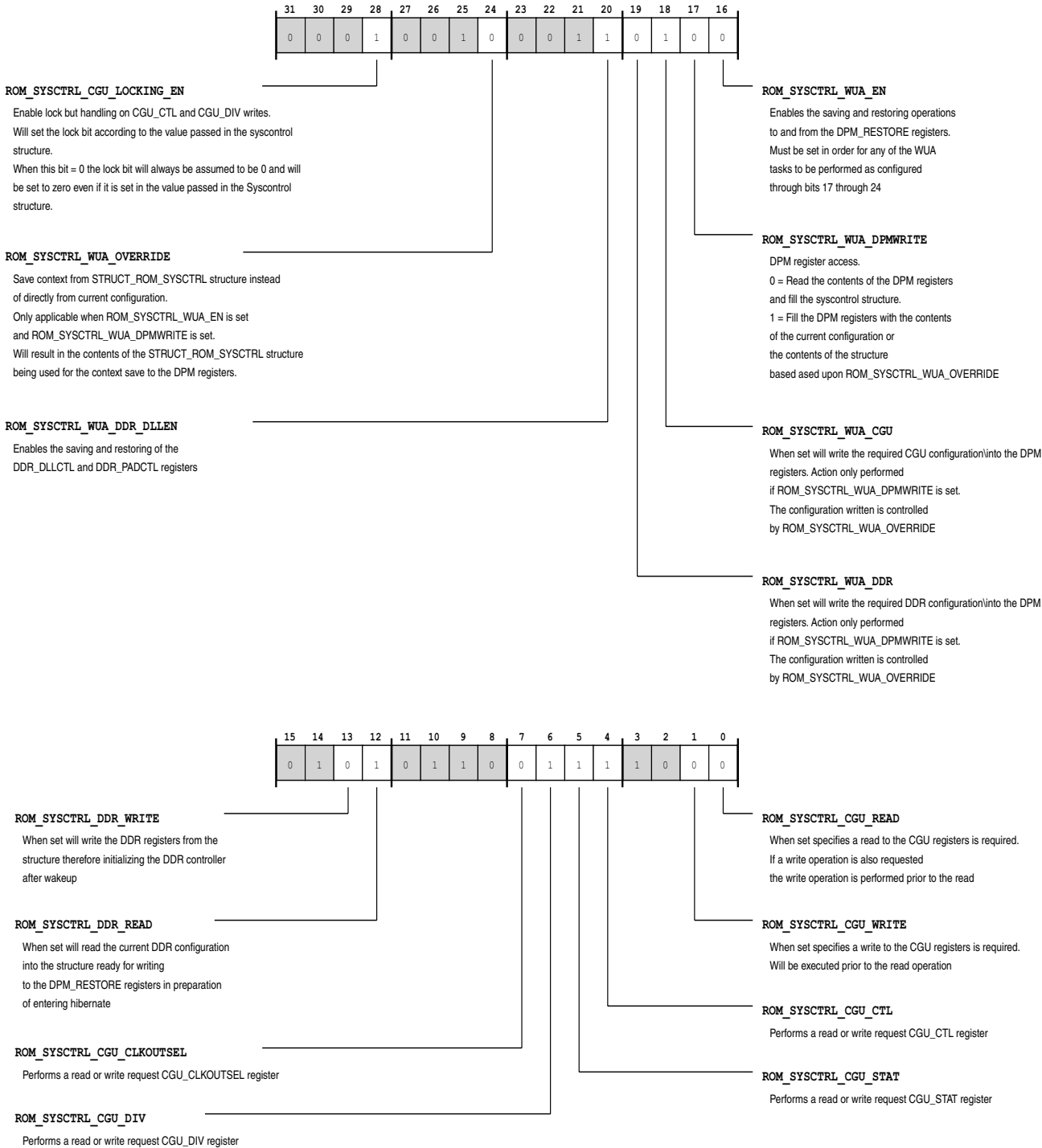


Figure 34-25: dActionFlags

## Boot Kernel

The boot kernel API can be used to implement custom boot modes.

```
void * rom_BootKernel(STRUCT_ROM_BOOT_CONFIG *pBootStruct);
```

### Boot Kernel

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| PP Define?         | FUNC_ROM_BOOTKERNEL                                                                                       |
| Prototype          | void * rom_BootKernel(STRUCT_ROM_BOOT_CONFIG *pBootStruct);                                               |
| Arguments?         | R0: Pointer to pBootStruct                                                                                |
| Return Value       | R0: pSource Pointer to next free source locations Pointer to DXE when invoked by ROM_BFLAG_NEXTDXE switch |
| Stack Requirements | valid stack (SP, FP dividable by 4) required                                                              |

## Boot Routine

The boot routine provides access to boot an application at run-time through a supported peripheral. The boot routine often also provides the ability for further customization over the equivalent boot mode.

```
rom_Boot(...)
```

The Boot Routine can be used for any kind of second-stage boot for supported boot modes. It provides options to boot from any device and any channel, whereas booting directly limits the choice to the default devices and channels. Often any auto-configuration or detection of the device can also be disabled. Each boot mode defines its own dBootCommand, (see individual boot modes for a description of this parameter). The \*pCallHook argument is only needed when BFLAG\_HOOK is set and should otherwise be NULL. The \*pTargetAddress argument is only needed when BFLAG\_DATAREAD is set and should otherwise be NULL.

|           |                                                                                                                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PP Define | FUNC_ROM_BOOT                                                                                                                                                                                                       |
| Prototype | void * rom_Boot( void *pBootStream, dFlags, int32_t dBlockCount, ROM_BOOT_HOOK_FUNC *pCallHook, uint32_t dBootCommand, void *pTargetAddress);                                                                       |
| Arguments | R0: Pointer to Boot Stream (memboot) or Start Address of Boot Stream (SPI boot)<br>R1: dFlags R2: dBlockCount<br>[fp+ 0x14] ROM_BOOT_HOOK_FUNC *pCallHook<br>[fp + 0x18] dBootCommand<br>[fp + 0x1C] pTargetAddress |

|                    |                                                                                                           |
|--------------------|-----------------------------------------------------------------------------------------------------------|
| Return Value       | R0: pSource Pointer to next free source locations Pointer to DXE when invoked by ROM_BFLAG_NEXTDXE switch |
| Stack Requirements | valid stack (SP, FP dividable by 4) required. 0x400 bytes of free stack suggested                         |

## CRC 32 Polynomial

Generates a CRC look-up table using CRC0.

```
bool rom_Crc32Poly(uint32_t ulPolynomial);
```

|                     |                                            |
|---------------------|--------------------------------------------|
| PP Define           | FUNC_ROM_CRC32POLY                         |
| Prototype?          | bool rom_Crc32Poly(uint32_t ulPolynomial); |
| Arguments?          | R0: Polynomial                             |
| Return Value        | 0: on success -1 on failure                |
| Stack Requirements? | none                                       |

## CRC Initcode

The CRC Initcode is called by the bootstream. A respective init block is inserted by the loader utility when invoked by the -crc switch. The CRC Initcode extracts the polynomial from the dArgument field of each block header and calls then the CRC LUT function.

```
void rom_Crc32Initcode( STRUCT_ROM_BOOT_CONFIG *pBootStruct);
```

|                    |                                                               |
|--------------------|---------------------------------------------------------------|
| PP Define?         | FUNC_ROM_CRC32INITCODE                                        |
| Prototype?         | void rom_Crc32Initcode( STRUCT_ROM_BOOT_CONFIG *pBootStruct); |
| Arguments?         | R0: Pointer to pBootStruct                                    |
| Return Value?      | none                                                          |
| Stack Requirements | valid stack (SP, FP dividable by 4) required.                 |

## ECC Protection

Protects 64-bit values by 8-bit ECC checksum using Hamming 7264 method. If return values equals ROM\_ECC\_ERROR\_1BIT source data is corrected in place. Cycle count is independent of data and whether correction takes place or not.

```
uint8_t rom_Ecc(u64 *pData, uint8_t bChksum, bool bDecode);
```

|                     |                                                                            |
|---------------------|----------------------------------------------------------------------------|
| PP Define?          | FUNC_ROM_ECC                                                               |
| Prototype?          | uint8_t rom_Ecc(u64 *pData, uint8_t bChksum, bool bDecode);                |
| Arguments           | R0: Pointer to 64-bit data<br>R1: 8-bit checksum<br>R2: 0=encode, 1=decode |
| Return Value?       | R0: 8-bit checksum when bDecode=false error code when dDecode=true         |
| Stack Requirements? | valid stack (SP, FP dividable by 4) required.                              |

**Table 34-22:** bDecode Values

| PP Define          | Description                     |
|--------------------|---------------------------------|
| ROM_ECC_ENCODE     | encode mode                     |
| ROM_ECC_DECODE     | decode mode                     |
|                    |                                 |
| ROM_ECC_ERROR_INVS | invalid syndrome/checksum       |
| ROM_ECC_ERROR_2BIT | uncorrectable 2-bit error       |
| ROM_ECC_ERROR_1BIT | 1-bit error, has been corrected |
| ROM_ECC_ERROR_NONE | no error                        |

## Execute

Resets the core as specified by dCoreId, stores dStartAddress into respective SVECTx registers and the releases the core.

|                    |                                                                        |                        |
|--------------------|------------------------------------------------------------------------|------------------------|
| Name               | Execute                                                                | -                      |
| PP Define          | FUNC_ROM_EXEC                                                          | -                      |
| Prototype          | void rom_Exec( uint32_t dCoreId, void* dStartAddress, void *reserved); | -                      |
| Argument           | R0                                                                     | 0=Core0, 1=Core1       |
| Argument           | R1                                                                     | Pointer to Routine     |
| Argument           | R2                                                                     | reserved, must be zero |
| Return Value       | none                                                                   | -                      |
| Stack Requirements | valid stack (SP, FP dividable by 4) required.                          | -                      |

```
rom_Exec(dCoreID,dStartAddress,0);
```

If dCoreId equals the current core,

- the routine simply issues a call to dStartAddress and returns.
- and the dStartAddress is 0x00000000 or 0xFFFFFFFF, the cycle counter is halted and an idle instruction is executed.

If dCoreId does not equal the current core,

- and the dStartAddress is 0x00000000, the target core is reset and kept in reset.
- and the dStartAddress is 0xFFFFFFFF, the target core vectors to ROM address where the cycle counter is halted and an idle instruction is executed.
- and the dStartAddress is 0xFFFFFFFEE, the target core's SVECTx is used without modification

For safety reasons, the rom\_Exec function does not unlock write access to the RCU0\_SVECT registers. Unlocking and re locking is expected to be managed by the calling function.

|           |                                                                        |
|-----------|------------------------------------------------------------------------|
| PP Define | FUNC_ROM_EXEC                                                          |
| Prototype | void rom_Exec( uint32_t dCoreId, void* dStartAddress, void *reserved); |

|                    |                                                                                   |
|--------------------|-----------------------------------------------------------------------------------|
| Arguments          | R0: 0=Core0, 1=Core1, ...<br>R1: Pointer to Routine<br>R2: reserved, must be zero |
| Return Value       | none                                                                              |
| Stack Requirements | valid stack (SP, FP dividable by 4) required.                                     |

## Forward Config

The Forward Config routine is provided initialize the link port and SPI peripherals for using the boot forward feature.

```
uint32_t* rom_ForwardConfig(STRUCT_ROM_BOOT_CONFIG *pBootStruct, uint32_t ulForwardPort);
```

The Forward Config routine initializes the link port and SPI peripherals for using the boot forward feature. This routine must be called from an initcode routine before any forward block can be processed. Refer to documentation on forward block (see [Block Types](#)) documentation for details on pBootStruct as the definition varies for each peripheral.

When called with the BFWD\_CLEANUP flag, the Forward Config routine clears the LPx\_CTL and DMay\_CONFIG registers.

|                     |                                                                                           |
|---------------------|-------------------------------------------------------------------------------------------|
| PP Define?          | FUNC_ROM_FWDCFG                                                                           |
| Prototype           | uint32_t* rom_ForwardConfig(STRUCT_ROM_BOOT_CONFIG *pBootStruct, uint32_t ulForwardPort); |
| Arguments?          | R0: Pointer to pBootStruct<br>R1: Forward Port details                                    |
| Return Value?       | R0: pointer to Forward Callback Function on success0 on error                             |
| Stack Requirements? | valid stack (SP, FP dividable by 4) required.                                             |

## Get Address

The Get Address routine can be used to access various look-up tables stored in the ROM. The function returns the address of the lookup table specified by the enum provided. Use this function rather than directly addressing tables to improve compatibility with future parts and silicon revisions.

## Functional Description

|                    |                                                                              |
|--------------------|------------------------------------------------------------------------------|
| PP Define          | FUNC_ROM_GETADDR                                                             |
| Prototype          | void * rom_GetAddr(enum ETABLES eTable);                                     |
| Arguments          | R0: eTable enumerator                                                        |
| Return Value       | R0: -1 in case of eTable was undefined enum start address of table otherwise |
| Stack Requirements | none                                                                         |

**Table 34-23:** eTable enumerators Values

| Index? | Returned Address           |
|--------|----------------------------|
| 0      | Global constants           |
| 1      | boot mode definition table |
| 2      | ecc syndrome table         |
| 3      | MDMACODE look-up table     |
| 4      | SPIMCODE look-up table     |
| 5      | RSICODE look-up table      |

## Mem Compare

Mem Compare compares a specified region of memory against a provide 32-bit reference value. The byte count can be any 32-bit value that is divisible by four, including zero. The DMA channel is disabled after completion, but the DMA status is not cleared so that in the event of an error the DMA can be debugged.

```
bool rom_MemCmp(void *pSrc, uint32_t ulChkVal, uint32_t ulByteCnt);
```

|            |                                                                     |
|------------|---------------------------------------------------------------------|
| PP Define? | FUNC_ROM_MEMCMP                                                     |
| Prototype? | bool rom_MemCmp(void *pSrc, uint32_t ulChkVal, uint32_t ulByteCnt); |



|                     |                                                                |
|---------------------|----------------------------------------------------------------|
| Arguments?          | R0: pSrc Source Address<br>R1: Compare Value<br>R2: Byte Count |
| Return Value        | 0: on success -1 on failure                                    |
| Stack Requirements? | none                                                           |

## Memory Copy

The Mem Copy function provides a convenient facility to copy memory using MDMA0 from one location to another. The byte count can be any 32-bit value including zero. The DMA channel is disabled after completion, but the DMA status is not cleared so that in the event of an error the DMA can be debugged.

```
bool rom_MemCpy(void *pDst, void *pSrc, uint32_t ulByteCnt);
```

|                     |                                                                           |
|---------------------|---------------------------------------------------------------------------|
| PP Define?          | FUNC_ROM_MEMCPY                                                           |
| Prototype?          | bool rom_MemCpy(void *pDst, void *pSrc, uint32_t ulByteCnt);              |
| Arguments           | R0: pDst Destination Address<br>R1: pSrc Source Address<br>R2: Byte Count |
| Return Value        | 0: on success -1 on failure                                               |
| Stack Requirements? | none                                                                      |

## Memory CRC

MemCRC scrubs memory using MDMA0 and CRC0 and builds a CRC checksum based on look-up table as generated by CRC32POLY beforehand. The byte count can be any 32-bit value that is divisible by four, including zero. The DMA channel is disabled after completion, but the DMA status is not cleared so that in the event of an error the DMA can be debugged.

```
bool rom_MemCrc(void *pSrc, uint32_t ulCrcChk, uint32_t ulByteCnt);
```

|           |                                                                     |
|-----------|---------------------------------------------------------------------|
| PP Define | FUNC_ROM_MEMCRC                                                     |
| Prototype | bool rom_MemCrc(void *pSrc, uint32_t ulCrcChk, uint32_t ulByteCnt); |
| Arguments | R0: pSrc Source Address<br>R1: Reference Checksum<br>R2: Byte Count |

|                    |                             |
|--------------------|-----------------------------|
| Return Value       | 0: on success -1 on failure |
| Stack Requirements | none                        |

## Memory Fill

Mem Fill fills a specified region of memory with a 32-bit value provided. The byte count can be any 32-bit value that is divisible by four, including zero. The DMA channel is disabled after completion, but the DMA status is not cleared so that in the event of an error the DMA can be debugged.

```
bool rom_MemFill(void *pDst, uint32_t ulFillVal, uint32_t ulByteCnt);
```

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| PP Define?          | FUNC_ROM_MEMFILL                                                      |
| Prototype?          | bool rom_MemFill(void *pDst, uint32_t ulFillVal, uint32_t ulByteCnt); |
| Arguments?          | R0: pDst DestinationAddress<br>R1: Fill Value<br>R2: Byte Count       |
| Return Value?       | 0: on success, -1 on failure                                          |
| Stack Requirements? | none                                                                  |

## Software Built-in Self Test

Provides entry to multiple Software-Based Functional Self-Testing routines.

```
ADI_SBST_RESULT rom_SBST(TEST_FUNC_ID, argument1, argument2, argument3);
```

|                     |                                                                                                                                                                            |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PP Define?          | FUNC_ROM_SBST                                                                                                                                                              |
| Address             | 0xC8000084                                                                                                                                                                 |
| Prototype?          | ADI_SBST_RESULT rom_SBST(TEST_FUNC_ID, argument1, argument2, argument3);                                                                                                   |
| Arguments?          | R0: ID of function to be executed<br>R1: Argument 1 to be passed to function<br>R2: Argument 2 to be passed to function<br>[fp+ 0x14]: argument 3 to be passed to function |
| Return Value?       | struct ADI_SBST_RESULT<br>R0: ADI_SBST_RESULT.dResult<br>R1: ADI_SBST_RESULT.dErrorAddress                                                                                 |
| Stack Requirements? | valid stack (SP, FP dividable by 4) required.                                                                                                                              |

## **Booting Data Structures**

The programming model for booting the processor uses the data structures defined in this section.

## STRUCT\_ROM\_SYSCTRL

Syscontrol configuration parameter

```
struct STRUCT_ROM_SYSCTRL;
```

|          |                |                                         |
|----------|----------------|-----------------------------------------|
| uint32_t | ulCGU_CTL      | Content of Clock Control Register       |
| uint32_t | ulCGU_DIV      | Content of Clock Divide Register        |
| uint32_t | ulWUA_Flags    | Flags used during wakeup                |
| uint32_t | ulWUA_BootAddr | Memory boot address                     |
| uint32_t | ulWUA_User     | User variable                           |
| uint32_t | ulDMC_CTL      | Content of the DMC Control Register     |
| uint32_t | ulDMC_CFG      | Content of the DMC Config Register      |
| uint32_t | ulDMC_TR0      | Content of the DMC Timing Register 0    |
| uint32_t | ulDMC_TR1      | Content of the DMC Timing Register 1    |
| uint32_t | ulDMC_TR2      | Content of the DMC Timing Register 2    |
| uint32_t | ulDMC_MR       | Content of the DMC MR Shadow Register   |
| uint32_t | ulDMC_EMR1     | Content of the DMC EMR1 Shadow Register |
| uint32_t | ulDMC_EMR2     | Content of the DMC EMR2 Shadow Register |
| uint32_t | ulDMC_PADCTL   | Content of the DMC PADCTL register      |
| uint32_t | ulDMC_DLLCTL   | Content of the DMC DLLCTL register      |
| uint32_t | ulReserved     | Reserved entry                          |

## STRUCT\_ROM\_BOOT\_BUFFER

```
struct STRUCT_ROM_BOOT_BUFFER
```

|         |            |
|---------|------------|
| void    | *pBuffer   |
| int32_t | dByteCount |

## STRUCT\_ROM\_BOOT\_CONFIG

```
struct STRUCT_ROM_BOOT_CONFIG
```

|                        |                      |
|------------------------|----------------------|
| void                   | *pSource             |
| void                   | *pDestination        |
| uint32_t               | *pControlRegister    |
| uint32_t               | *pAuxControlRegister |
| uint32_t               | *pDmaControlRegister |
| uint32_t               | *pSecControlRegister |
| int32_t                | dControlValue        |
| int32_t                | dByteCount           |
| int32_t                | dFlags               |
| uint16_t               | uwDataWidth          |
| uint16_t               | uwSrcModifyMult      |
| uint16_t               | uwDstModifyMult      |
| uint16_t               | uwUserShort          |
| int32_t                | dUserLong            |
| int32_t                | dReserved            |
| void                   | *pModeData           |
| int32_t                | dBootCommand         |
| void                   | *pNextDxe            |
| ROM_BOOT_ERROR_FUNC    | *pErrorFunction      |
| ROM_BOOT_LOAD_FUNC     | *pLoadFunction       |
| ROM_BOOT_CALLBACK_FUNC | *pCallBackFunction   |

|                        |                   |
|------------------------|-------------------|
| ROM_BOOT_CALLBACK_FUNC | *pCrcFunction     |
| ROM_BOOT_CALLBACK_FUNC | *pForwardFunction |
| STRUCT_ROM_BOOT_HEADER | *pHeader          |
| void                   | *pTempBuffer      |
| int32_t                | dTempByteCount    |
| void                   | *pTempSource      |
| int32_t                | dPageByteCount    |
| uint32_t               | ulBlockCount      |
| uint32_t               | ulBlockCurrent    |
| int32_t                | dClock            |
| void                   | *pLogBuffer       |
| void                   | *pLogCurrent      |
| int32_t                | dLogByteCount     |

## STRUCT\_ROM\_BOOT\_HEADER

```
typedef struct STRUCT_ROM_BOOT_HEADER
```

|         |                 |
|---------|-----------------|
| int32_t | dBlockCode      |
| void    | *pTargetAddress |
| int32_t | dByteCount      |
| int32_t | dArgument       |

## STRUCT\_ROM\_BOOT\_SPI

```
struct STRUCT_ROM_BOOT_SPI
```

|          |                |
|----------|----------------|
| uint8_t  | ubReadCommand  |
| uint8_t  | ubDummyBytes   |
| uint8_t  | ubAddressBytes |
| uint8_t  | ubDataBits     |
| uint16_t | uwClkLower     |
| uint16_t | uwTxCtlUpper   |
| uint16_t | uwRxCtlUpper   |
| uint16_t | uReserved      |

## Wakeup From Hibernate

When the boot code detects that the chip is waking up from hibernate it can take some special actions. These include:

- Configuration of PLL (CGU\_DIV and CGU\_CTL registers)
- Initialization of DDR controller

- Memboot from any memory address

The special sequence executes when the RCU\_HBTRES hibernate reset identification bit is set. In case of a warm boot the RCU\_HBTRES bit can be overruled by the BCODE\_HBTOVW bit. When either of these bits is detected as a one, the boot code inspects the DPM\_RESTORE0 register for further instructions. At least two bits must be set in DPM\_RESTORE0 for the special action to take place.

Note the importance of this feature in case of slave boot modes. The Blackfin enters hibernate on its own decision. There is no direct path for this action to be directly controlled by the host (it might indirect thought). Therefore, do not assume that the host was aware of the processor being in the hibernate state.

On wakeup, the host may not know that the processor is expecting boot data. To prevent the processor from starving at wakeup, this special functionality enables the processor to boot from a LDR stream stored in off-chip memory (DDR or SMC SRAM). The stream may consist of one single 16-byte boot block header, which efficiently terminates the boot process and enables Core 0 to execute the code directly without further involvement of the boot code. Since even this one boot header is protected by XOR checksum the processor is prevented from booting illegal opcodes when the memory content was not in integer form.

When the WUA\_MEMBOOT and WUA\_ENA bits are set, after optional PLL and DDR initialization, the boot code does not continue with boot mode processing as per the BMODE pin configuration. Instead, the boot code performs a memory boot from the address specified by the WUA\_BOOT\_ADDR field. In case the address points to SMC space the memory boot mode automatically enables the respective SMC bank and required pin muxing.

## **CGU Initialization after Wakeup**

When the processor enters the hibernate state, the CGU current configuration is lost and at a wakeup event the default reset values become applicable. In order to provide more optimal processor configuration at wakeup, programs may save the required CGU configuration to the DPM0\_RESTORE registers. Upon detecting a wakeup event the boot code can optionally re-configure the CGU based on the contents of the DPM0\_RESTORE registers.

The WUA\_CLOCK bit of the DPM0\_RESTORE0 register defines whether the boot code should reconfigure the CGU or not. If the CGU is to be reconfigured then the CGU0\_DIV and the CGU0\_CTL registers are programmed based upon the contents of two of the DPM0\_RESTORE registers. See the registers section below for details of which registers are used during the configuration.

## **DDR Controller Initialization after Wakeup**

Before entering the hibernate state, the external SDRAM is placed in self-refresh mode in order to preserve the contents of memory. During hibernate the power to the DDR is cut-off, to lower power consumption. When the device exits the hibernate state the DDR is required to be re-initialized. In order for the DDR to be re-initialized, programs are required to save the current DDR configuration to the DPM\_RESTORE registers prior to the processor entering the hibernate state. The Syscontrol routine provides a means for



programs to restore the context prior to entering hibernate in a manner that is compatible with the required context restoration.

## BFLAG\_WAKEUP and BFLAG\_QUICKBOOT

Whenever either the RCU\_BCODE\_HBTOVW flag or the RCU\_STAT\_HBRST status bit is set, the main routine passes the BFLAG\_WAKEUP flag when calling the Boot routine. When called that way the underlying boot kernel watches for block headers for the BFLAG\_QUICKBOOT flag set. In the event of both flags being set, the boot kernel takes special action in the scope of the current block by:

- Toggling the BFLAG\_IGNORE flag
- Clearing the BFLAG\_INIT, BFLAG\_CALLBACK, BFLAG\_FORWARD, BFLAG\_AUX and BFLAG\_FINAL flags (as with any ignore block)
- Prevents fill blocks from processing (as with any ignore block)

Blocks that have BFLAG\_IGNORE=0 and BFLAG\_QUICKBOOT=1 are normally processed for a regular (non-wakeup) boot and are not processed for a wakeup boot. This way unnecessary rebooting of non-volatile memory cells can be avoided (speak external SRAM or SDRAM in self-refresh mode).

Blocks that have BFLAG\_IGNORE=1 and BFLAG\_QUICKBOOT=1 are ignored during regular boot and become active in case of wakeup. This supports complete arbitrate boot processing. Note that for FIRST blocks the IGNORE flag only controls the data load. The handling of the start vector and the next dx pointer is not conditional. To make the initcode processing conditional, keep the BFLAG\_QUICKBOOT flag clear for the init block and interrogate the BFLAG\_WAKEUP flag in the initcode itself.

The BFLAG\_SAVE flag is the BFLAG\_QUICKBOOT's counterpart and enables programs to implement special context savings strategies before entering hibernate. However, the boot code does not directly support this functionality.

The wakeup/quickboot mechanism can be re-purposed by controlling the BFLAG\_WAKEUP bit in the rom\_Boot() routine's dFlags argument, or by altering the flag from within hook or initcode routines.

## DPM Register List

The dynamic power management (DPM) unit includes the phase locked loop (PLL) enable/disable features, deep sleep and hibernate mode controls, and clock domain enable/disable features. The combination of these features and controls provide selective and flexible power management. A set of registers govern DPM operations. For more information on DPM functionality, see the DPM register descriptions.

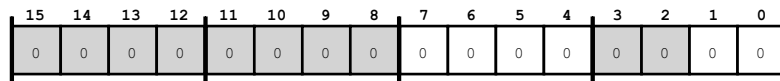
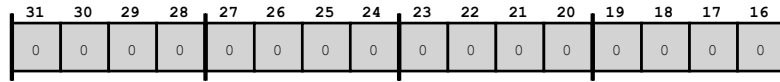
**Table 34-24:** DPM Module Registers

| Name               | Description                              |
|--------------------|------------------------------------------|
| DPM_CTL            | Control Register                         |
| DPM_STAT           | Status Register                          |
| DPM_CCBF_DIS       | Core Clock Buffer Disable Register       |
| DPM_CCBF_EN        | Core Clock Buffer Enable Register        |
| DPM_CCBF_STAT      | Core Clock Buffer Status Register        |
| DPM_CCBF_STAT_STKY | Core Clock Buffer Status Sticky Register |
| DPM_SCBF_DIS       | System Clock Buffer Disable Register     |
| DPM_WAKE_EN        | Wakeup Enable Register                   |
| DPM_WAKE_POL       | Wakeup Polarity Register                 |
| DPM_WAKE_STAT      | Wakeup Status Register                   |
| DPM_HIB_DIS        | Hibernate Disable Register               |
| DPM_PGCNTR         | Power Good Counter Register              |
| DPM_RESTOREn       | Restore n Register                       |

## DPM Restore

### DPM\_RESTORE0: DPM Restore Register 0 (R/W, 3V domain) - RW

Reset = 0x0000 0000



#### WUA\_DDR\_LOCK

Action on Wakeup from Hibernate

0 = Waits for (DSEL/CSEL)+1\*3000 CCLK cycles for DLL lock to occur

1 = Waits for (DSEL/CSEL)+1\*6000 CCLK cycles for DLL lock to occur

#### WUA\_DDR\_DLLEN

Action on Wakeup from Hibernate

0 = Do not restore DDR\_DLLCTL and DDR\_PADCTL registers

1 = Restore DDR\_DLLCTL and DRR\_PADCTL registers

#### WUA\_DDR

Action on Wakeup from Hibernate

0 = Do not initialize DDR controller

1 = Initialize DDR controller

#### WUA\_CLOCK

Action on Wakeup from Hibernate

0 = Do not initialize from CGU

1 = Initialize CGU

#### WUA\_ENA

Wakeup Sequencing Enable bit

0 = No special action is taken

1 = Special action is taken based on further instruction bits in this register

#### WUA\_CLOCK

Action on Wakeup from Hibernate

0 = Boot from regular BMODE source

1 = Boot from user address in memory

**Figure 34-26:** DPM Restore Register0

The following table describes the function of each of the restore registers. RESTORE0 and RESTORE1 are only registers that do not directly restore other registers. RESTORE0 serves as the control register, enabling the function of the rest, and RESTORE1 stores a boot address rather than data to restore a particular register.

**Table 34-25:** DPM Restore Values

| DPM Register | Restored Register   | Enable Bit  |
|--------------|---------------------|-------------|
| RESTORE1     | Address booted from | WUA_MEMBOOT |
| RESTORE2     | CGU_DIV             | WUA_CLOCK   |
| RESTORE3     | CGU_CTL             | WUA_CLOCK   |
| RESTORE4     | Reserved            | Reserved    |
| RESTORE5     | DDR_CTL             | WUA_DDR     |

**Table 34-25:** DPM Restore Values (Continued)

| DPM Register | Restored Register | Enable Bit             |
|--------------|-------------------|------------------------|
| RESTORE6     | DDR_CFG           | WUA_DDR                |
| RESTORE7     | DDR_TR0           | WUA_DDR                |
| RESTORE8     | DDR_TR1           | WUA_DDR                |
| RESTORE9     | DDR_TR2           | WUA_DDR                |
| RESTORE10    | DDR_MR            | WUA_DDR                |
| RESTORE11    | DDR_EMR1          | WUA_DDR                |
| RESTORE12    | DDR_EMR2          | WUA_DDR                |
| RESTORE13    | DDR_PADCTL        | WUA_DDR& WUA_DDR_DLLEN |
| RESTORE14    | DDR_DLLCTL        | WUA_DDR& WUA_DDR_DLLEN |
| RESTORE15    | Reserved          | Reserved               |

## Reset and Power-up

The processor implements several different power-up and reset scenarios ranging from a full power-on to software triggered resets. The Boot Kernel plays a specific role depending on the scenario.

### Reset Vector

When reset releases, the processor starts fetching and executing instructions at the on-chip boot ROM.

On a hardware reset the boot kernel initializes the `RCU_SVECT` register to `0xFFA00000`. When the booting process completes, the boot kernel jumps the location provided by the `RCU_SVECT` vector register. With the exception of the host DMA boot modes, the content of the `RCU_SVECT` register is overwritten by the target address field of the first block of the loaded boot stream. If the `BCODE` field of the `RCU_STAT` register is set to No-boot, the `RCU_SVECT` register is not modified by the boot kernel on software resets. Therefore, programs can control the reset vector for software resets through the `RCU_SVECT` register. The content of the register may be `RCU_SVECT` undefined in emulator sessions.

## Servicing Reset Interrupts

The reset interrupt has top priority and the processor services a reset event like other interrupts. Only emulation events have higher priority. When it comes out of reset, the processor is in supervisor mode and has full access to all system resources. The boot kernel can be seen as part of the reset service routine as it runs at top interrupt priority level.

Even when the boot process has finished and the boot kernel passes control to the user application, the processor is still in the reset interrupt. To enter the user mode, the reset service routine must initialize the RETI register and terminate by an RTI instruction.

Systems that do not work in an OS environment may not enter user mode. Typically, the interrupt level needs to be degraded down to IVG15.

**NOTE:** As the boot kernel is running at the reset interrupt priority,  $\overline{\text{NMI}}$  events, hardware errors, and exceptions are not served at boot time. As soon as the reset service routine returns, the processor may service the events that occurred during the boot sequence. It is recommended that programs install  $\overline{\text{NMI}}$ , hardware error and exception handlers before leaving the reset service routine. This includes proper initialization of the respective event vector registers, RCU\_SVECT

## NMI and RESOUT

The  $\overline{\text{NMI}}$  and  $\overline{\text{RESOUT}}$  signals share the same pin. When the part is in reset, the pin is in  $\overline{\text{RESOUT}}$  mode, and  $\overline{\text{RESOUT}}$  is asserted. The  $\overline{\text{NMI}}$  signal is disabled for the duration of this reset event. After the  $\overline{\text{RESOUT}}$  goes low the pin is automatically placed into  $\overline{\text{NMI}}$  mode and the  $\overline{\text{NMI}}$  signal is enabled.

This pin should be connected to an external pull-up.

## Memory Initialization

The first stage of booting initializes and validates all memory. This is performed before the actual boot process begins.

Memory Initialization occurs early in the boot process, before any stream information is read, and before the boot code has any influence. The boot kernel initializes all memory and cache tags and ensures all parity and ECC checksum bits are consistent with data. All memory is initialized to 0x00A700A7 or except 7;

Both cores are initialized. This infers that both cores will be woken up during this initial boot process. When booting from Core 0, upon completion of memory initialization, core 1 is left in the idle state. Core 1 can be used during the boot process using the API function.

## Debugging

This section describes how to bypass booting and execute code directly from SDRAM

If the boot process fails, very little information can be gained by watching the chip from outside. In master boot modes, the interface signals can be observed. In slave boot modes, only the `RESOUT` and `RTS` signals tell about the progress of the boot process.

The `RESOUT` signal is cleared after memory initializations are completed. The `RESOUT` and `NMI` signals share the same pin. During the boot process the `NMI` signal is disabled, once the `RESOUT` signal is asserted high upon completion of the boot process, the `NMI` signal is enable. Note that the `RESOUT` and `NMI` signals are active low and therefore require a pull-up resistor to be connected.

Using the emulator, however, there are many possibilities to debug the boot process. The entire source code of the boot kernel is provided with the tools installation. This includes the DSP executable (DXE) file. The *load symbols* feature of the tools environment helps to navigate the program. Note that the content of the ROM might differ between silicon revisions. Hardware breakpoints and single-stepping capabilities are also available. Since the content of the L1 instruction ROM cannot be read out by the emulator, as this ROM is not supported by the ITEST feature, these instructions are not displayed in the disassembly window. Table [Boot Kernel Symbol for Debug](#) shows several program symbols that may be of interest.

**Table 34-26:** Boot Kernel Symbol for Debug

| Symbol                                      | Comment                                                                                                                                                                                                                                        |
|---------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>_bootrom.bootmenu</code>              | If the emulator hits a hardware breakpoint at the <code>_bootrom.bootmenu</code> address, this indicates that the preboot returned properly. Otherwise the program may hang during preboot due to improper PBS settings or invalid boot modes. |
| <code>_bootrom.bootkernel.entry</code>      | If the emulator hits a hardware breakpoint at the <code>_bootrom.bootkernel.entry</code> label, this indicates that device detection or autobaud returned properly.                                                                            |
| <code>_bootrom.bootkernel.breakpoint</code> | This is a good address to place a hardware breakpoint. When hit the boot kernel has a new block header loaded in. The block header can be watched at address wherever the <code>pHeader</code> points to.                                      |
| <code>_bootrom.bootkernel.initcode</code>   | All payload data of the current block has been loaded by the time the program passes this label. The boot kernel is about to interrogate the <code>BFLAG_INITCODE</code> flag. If set, the <code>initcode</code> can be debugged.              |
| <code>_bootrom.bootkernel.exit</code>       | Once the boot kernel arrives at this label, it has detected a <code>BFLAG_FINAL</code> flag. After some housekeeping, it jumps to the <code>EVT1</code> vector.                                                                                |

The boot kernel also generates a circular log file in scratch pad memory. While the `pLogBuffer` and the `dLogByteCount` variables describe the location and dimension of the log buffer, the `pLogCurrent` points to the next free location in the buffer. The log file is updated whenever the kernel passes the `_bootrom.bootkernel.breakpoint` label.

At each pass, nine 32-bit words are written to the log file, as follows.

- the block code word (dBlockCode) of the block header
- the target address (dTargetAddress) of the block header
- the byte count (dByteCount) of the block header
- the argument word (dArgument) of the block header
- the source pointer (dSource) of the boot stream
- the block count (dBlockCount)
- an internal copy of the dBlockCode word OR'ed with dFlags
- the content of the SEQSTAT register
- a 0xFFFFFFFFFA constant

The ninth word is overwritten by the next entry set, so that 0xFFFFFFFFFA always marks the last entry in the log file. Most of the data structures used by the boot kernel reside of the stack in scratch pad memory. While executing the boot kernel routine (excluding subroutines) the P5 point to the *STRUCT\_ROM\_BOOT\_CONFIG* structure. Type *STRUCT\_ROM\_BOOT\_CONFIG\**) \$P5 in the IDDE expression window to see the structure content.

The *Error Handler* can also give indications of errors. See the API documentation for more information.

## Boot ROM Revision Control

Describes the provisions for reading ROM version on the processor

### Boot ROM Revision Control

The boot ROM reserves the 32-bit location at address 0xC8000040 for a version code consisting of four bytes as shown in the *ROM Revision Control*.

Revision: Memory location with ROM revision information -

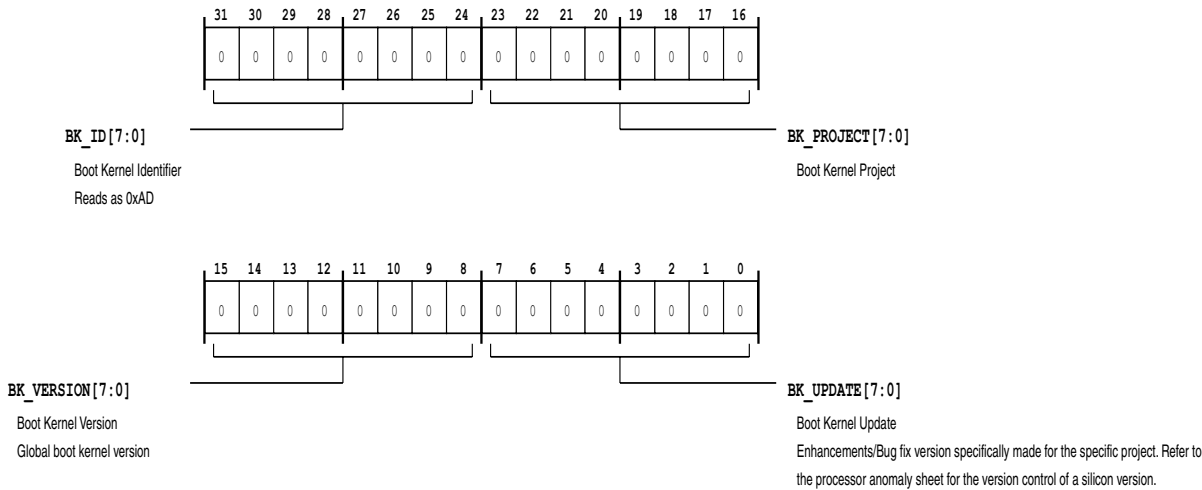


Figure 34-27: ROM Revision Control

# Booting Register Reference

This section provides reference information regarding the registers used during the booting process.

## Status Register

The RCU status register (RCU\_STAT) contains status bits for all RCU reset sources, reset status, and boot mode inputs. Status bits for reset sources are sticky and can be cleared by software. Error status bits are cleared by any reset event.



**RCU\_STAT: Status Register - R/W**

Reset = 0x0000 0021

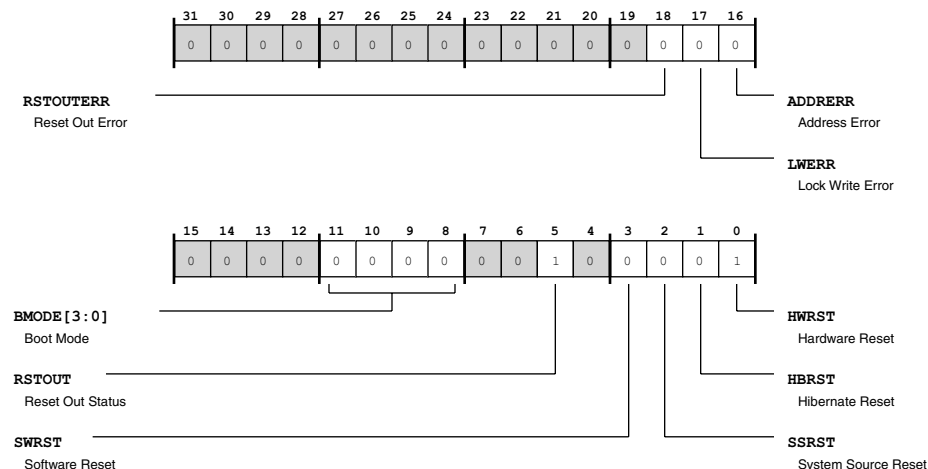


**Figure 34-28:** RCU\_STAT Register Diagram

**Table 34-27:** RCU\_STAT Register Fields

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                                                           |
|---------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | RSTOUTERR | Reset Out Error.<br>The RCU_STAT . RSTOUTERR bit indicates (if set) that a write attempted to set the RCU_CTL . RSTOUTASRT and RCU_CTL . RSTOUTDSRT simultaneously. This condition triggers a bus error.                                                          |
|         |           | 0 No Error                                                                                                                                                                                                                                                        |
|         |           | 1 Error Occurred                                                                                                                                                                                                                                                  |
| 17      | LWERR     | Lock Write Error.<br>The RCU_STAT . LWERR bit indicates (when set) there was an attempted write to an RCU register while the the RCU_CTL . LOCK bit was set and the global lock bit is enabled (SPU_CTL_GLCK bit =1). This status bit is sticky; write-1-to-clear |
|         |           | 0 No Error                                                                                                                                                                                                                                                        |
|         |           | 1 Error Occurred                                                                                                                                                                                                                                                  |

**Table 34-27:** RCU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | ADDRERR  | Address Error.<br>The RCU_STAT.ADDRERR bit indicates that the RCU generated an address error. This status bit is sticky; write-1-to-clear it.                                                                                                                                        |
|         |          | 0 No Error                                                                                                                                                                                                                                                                           |
|         |          | 1 Error Occurred                                                                                                                                                                                                                                                                     |
| 11:8    | BMODE    | Boot Mode.<br>The RCU_STAT.BMODE bits indicate the input on the boot mode pins.                                                                                                                                                                                                      |
| 5       | RSTOUT   | Reset Out Status.<br>The RCU_STAT.RSTOUT bit indicates the assertion status of the system reset pin.                                                                                                                                                                                 |
|         |          | 0 RSTOUT Deasserted                                                                                                                                                                                                                                                                  |
|         |          | 1 RSTOUT Asserted                                                                                                                                                                                                                                                                    |
| 3       | SWRST    | Software Reset.<br>The RCU_STAT.SWRST bit indicates that a system reset (which was triggered by software) has occurred since the last time a hardware reset occurred or since the RCU_STAT.SWRST bit was cleared by software.                                                        |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |
| 2       | SSRST    | System Source Reset.<br>The RCU_STAT.SSRST bit indicates that a system reset triggered by hardware in the system clock domain, clock A domain, or clock B domain has occurred since the last time a hardware reset occurred or since the RCU_STAT.SSRST bit was cleared by software. |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                           |
|         |          | 1 Reset Occurred                                                                                                                                                                                                                                                                     |

**Table 34-27:** RCU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                   |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | HBRST    | Hibernate Reset.<br>The <code>RCU_STAT.HBRST</code> bit indicates that a hibernate reset has occurred since the last time a hardware reset occurred or since the <code>RCU_STAT.HBRST</code> bit was cleared by software. |
|         |          | 0 Inactive                                                                                                                                                                                                                |
|         |          | 1 Reset Occurred                                                                                                                                                                                                          |
| 0       | HWRST    | Hardware Reset.<br>The <code>RCU_STAT.HWRST</code> bit indicates that a hardware reset has occurred.                                                                                                                      |
|         |          | 0 Inactive                                                                                                                                                                                                                |
|         |          | 1 Reset Occurred                                                                                                                                                                                                          |

## Software Vector Register n

The RCU software vector n registers (`RCU_SVECTn`) each contain a software vector (reset vector) for the processor cores.

**RCU\_SVECTn: Software Vector Register n - R/W**

Reset = 0x0000 ffa0

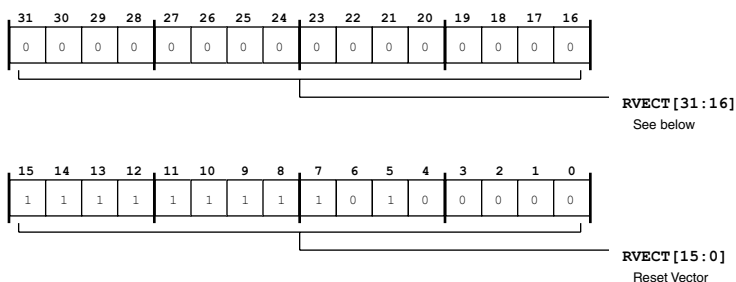


**Figure 34-29:** RCU\_SVECTn Register Diagram

**Table 34-28:** RCU\_SVECTn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | RVECT    | Reset Vector.           |

## Boot Code Register

The RCU software vector lock register (RCU\_BCODE) provides a register lock and software vector n enable bits for each processor core on the product.

RCU\_BCODE: Boot Code Register - R/W

Reset = 0x0000 0000

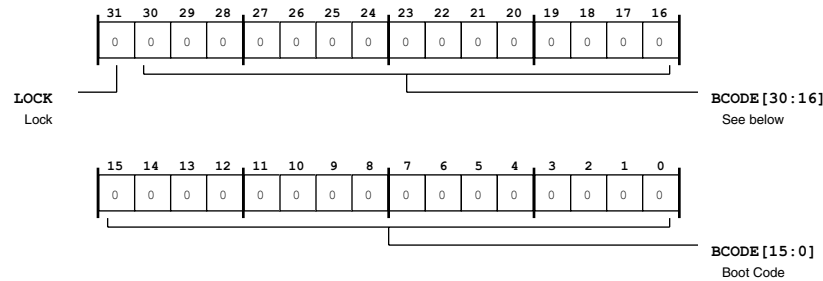


Figure 34-30: RCU\_BCODE Register Diagram

Table 34-29: RCU\_BCODE Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                         |        |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| 31      | LOCK     | Lock.<br>If the global lock bit is set (SPU_CTL_GLCK bit =1) and the RCU_BCODE . LOCK bit is set, the RCU_BCODE register is read only (locked). |        |
|         |          | 0                                                                                                                                               | Unlock |
|         |          | 1                                                                                                                                               | Lock   |
| 30:0    | BCODE    | Boot Code.<br>The RCU_BCODE . BCODE bits contain a boot code for the processor. For more information, see the RCU functional description.       |        |

# 35 System Debug Unit (SDU)

The System Debug Unit (SDU) provides debug access to the connected system. It provides debug host interface support through a JTAG (IEEE-1149.1) interface. In addition to traditional JTAG features, the SDU provides direct access to system resources to enable highly flexible and non intrusive debug support. The SDU can control and communicate with cores through JTAG or (if supported by the core) through system transactions. The SDU can perform system transactions as a system master through the memory access controller interface (MAC) and its memory mapped registers can be accessed by system masters through the slave port.

## SDU Features

- System JTAG TAP controller for system debug features, boundary scan, and public JTAG features
- Provides debug interface to core(s), and other system resources
- Provides direct and runtime access to memory system and system MMRs
- Provides DMA and non-DMA access to memory spaces
- SDU registers accessible to debug host and system (system MMRs)
- Support for system watchpoint and other event signaling
- Direct control over system reset
- Provides support for debug immediately after reset (boot debug)
- Supports group halt (debug event immediately halts all specified endpoints)

## SDU Functional Description

The following sections provide a functional description of the SDU.

- [\*SDU Register List\*](#)
- [\*ADSP-BF60x SDU Interrupt List\*](#)
- [\*SDU Block Diagram\*](#)
- [\*JTAG Tap Controller \(JTC\) Block Diagram\*](#)
- [\*Memory Access Controller \(MAC\)\*](#)

- [Group Halt](#)

## SDU Register List

The system debug unit (SDU) provides IEEE-1149.1 support through its JTAG interface. The registers listed in the SDU register summary table govern SDU operations. For more information on SDU functionality, see the SDU register descriptions.

**Table 35-1:** SDU Module Registers

| Name        | Description                    |
|-------------|--------------------------------|
| SDU_IDCODE  | ID Code Register               |
| SDU_CTL     | Control Register               |
| SDU_STAT    | Status Register                |
| SDU_MACCTL  | Memory Access Control Register |
| SDU_MACADDR | Memory Access Address Register |
| SDU_MACDATA | Memory Access Data Register    |
| SDU_DMARD   | DMA Read Data Register         |
| SDU_DMAWD   | DMA Write Data Register        |
| SDU_MSG     | Message Register               |
| SDU_MSG_SET | Message Set Register           |
| SDU_MSG_CLR | Message Clear Register         |
| SDU_GHLT    | Group Halt Register            |

## ADSP-BF60x SDU Interrupt List

**Table 35-2:** ADSP-BF60x SDU Interrupt List

| Description | ID | DMA Channel | Sensitivity |
|-------------|----|-------------|-------------|
| SDU0 DMA    | 64 | 11          | LEVEL       |

## ADSP-BF60x SDU Trigger List

**Table 35-3:** Masters ADSP-BF60x SDU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
| SDU0 DMA    | 31 | PULSE/EDGE  |
|             |    |             |

**Table 35-4:** Slaves ADSP-BF60x SDU Trigger List

| Description        | ID | Sensitivity |
|--------------------|----|-------------|
| SDU0 DMA           | 31 |             |
| SDU0 Slave Trigger | 69 |             |
|                    |    |             |

## SDU Definitions

To make the best use of the SEC, it is useful to understand the following terms.

### *JTAG 1149.1*

IEEE Standard Test Access Port and Boundary-Scan Architecture.

### **Group Halt**

Synchronous halt of distributed bus masters and various peripherals.

### **MAC**

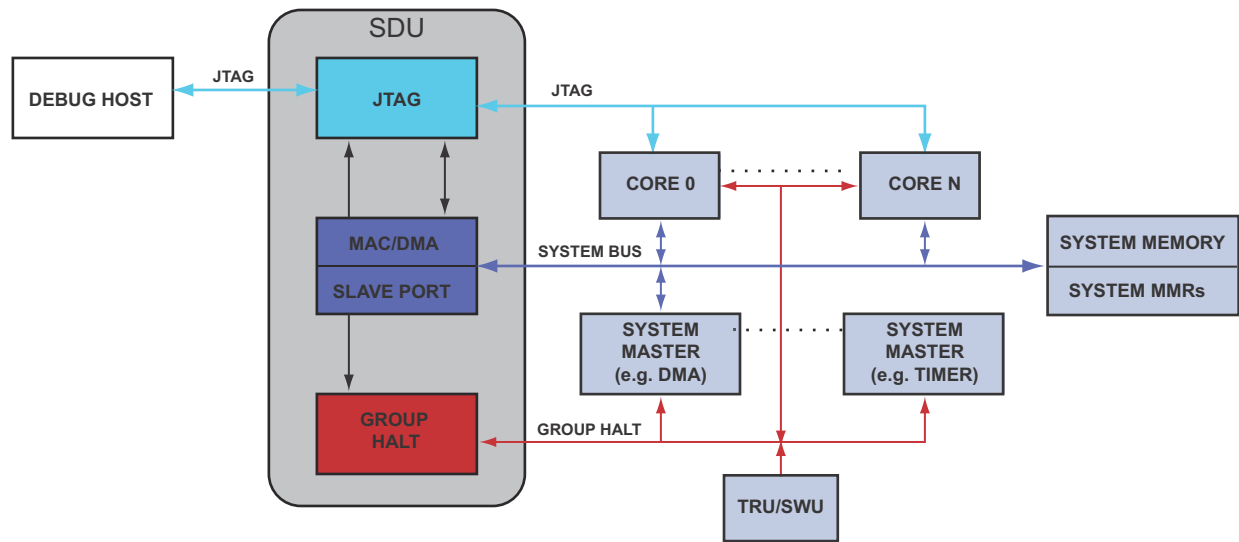
Memory Access Controller, system master interface controlled by the SDU

### **System Bus**

In the context of the SDU definition this refers to the system interconnect fabric of the given product.

## SDU Block Diagram

The SDU architectural model is illustrated in the following figure. The figure is a block diagram of the SDU and the associated debug connections of the system.



## JTAG Tap Controller (JTC) Block Diagram

The SDU includes a JTAG tap controller to support the IEEE-1149.1 standard. The JTAG tap controller implements the traditional state machine for the TAP controller and the necessary support for the implemented scan chains. The figure below shows the JTAG (IEEE1149.1) TAP controller state machine.



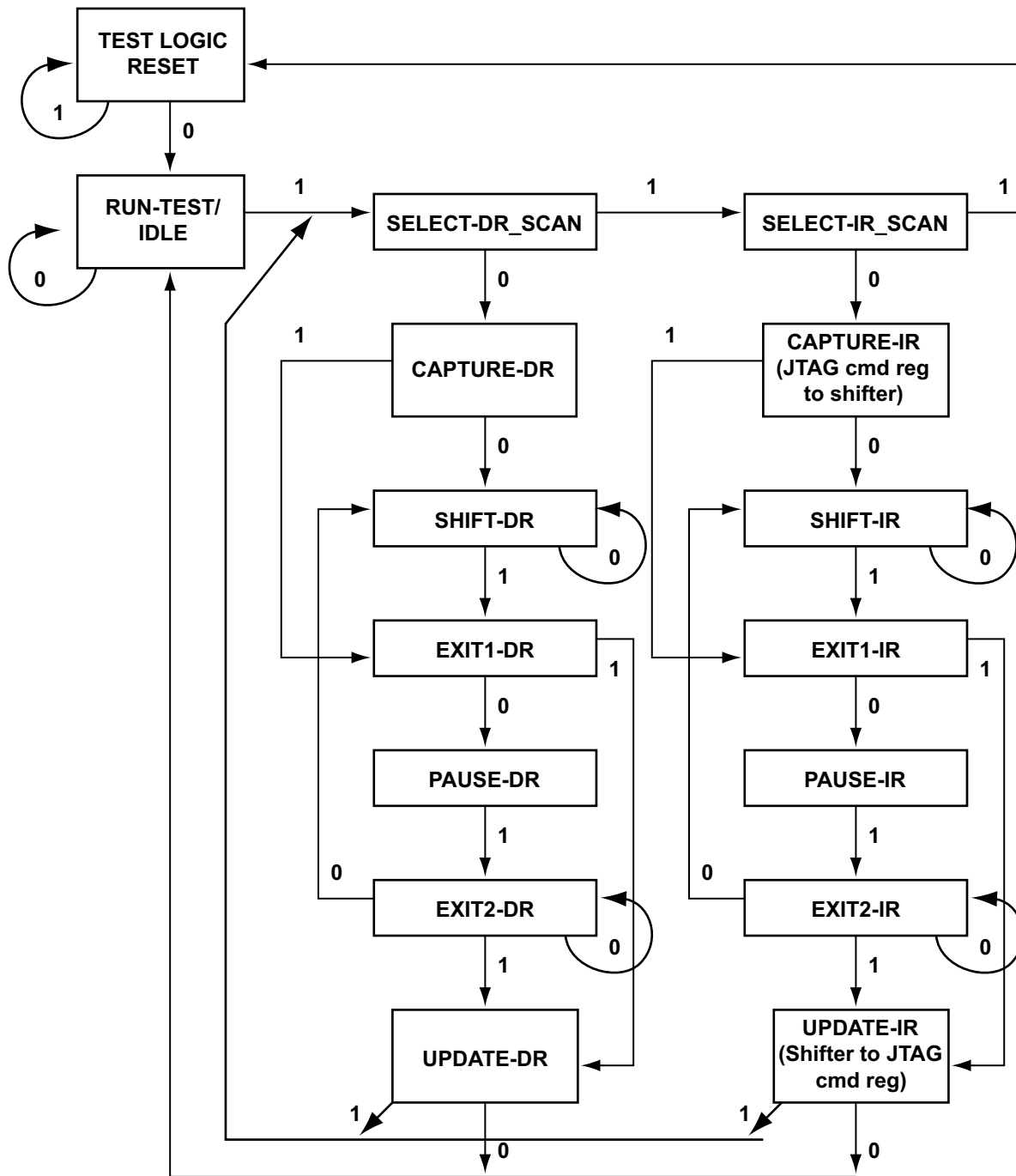
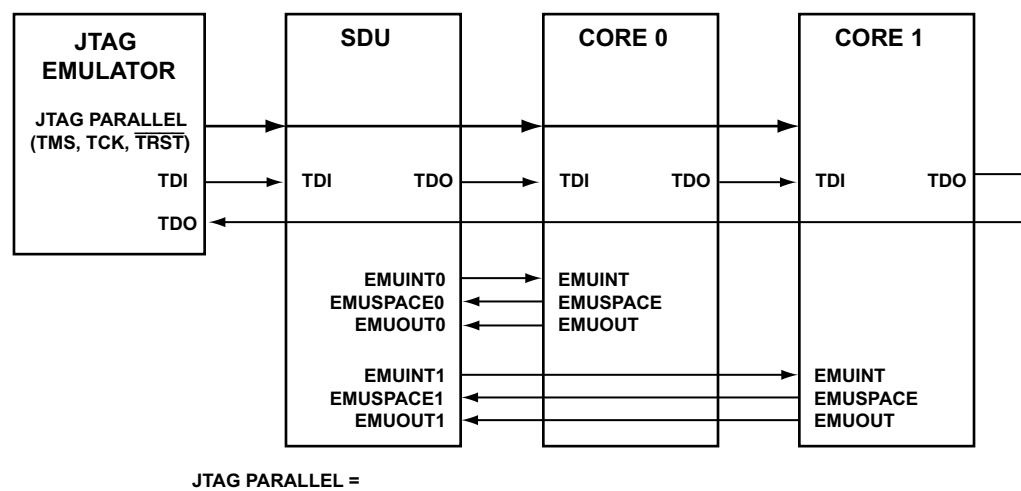


**Figure 35-1:** JTAG TAP Controller

## JTC Core Emulation

Each core has its own JTAG tap controller in addition to the JTC. The core JTAG tap controllers are bypassed (not in the scan path) after reset. They can be added to the scan path under control of the SDU\_CTL register. Each core JTAG controller provides support for the core only feature set (step, core register access and others). The following figure illustrates the connection model.



JTC Instruction Register (IR)

The Instruction Register of the JTC is five bits wide. Decode of the Instruction Register selects which of the SDU scan registers is to be connected between TDI and TDO pins during scan operation. The SDU scan registers provide support for JTAG Public Instructions and private (SDU) registers for debug control and status.

The following tables list the JTAG scan registers with their corresponding instruction decode. Instruction decode values not listed are reserved and must not be accessed.

Table 35-5: JTAG Public Scan Registers

| Instruction Decode (TDI->TDO) | Instruction Names | Register Name                              | Width            |
|-------------------------------|-------------------|--------------------------------------------|------------------|
| 00000                         | Extest            | Boundary Scan                              | Product Specific |
| 10000                         | Sample/Preload    | Boundary Scan                              | Product Specific |
| 00010                         | IDCODE            | CHIPID - Constant (Product Specific) value | 32               |
| 11111                         | Bypass            | Bypass                                     | 1                |

Table 35-6: SDU Scan Registers

| Instruction Decode (TDI->TDO) | Instruction Names | Register Name        | Width |
|-------------------------------|-------------------|----------------------|-------|
| 00101                         | SDU_CTL           | SDU Control Register | 16    |

**Table 35-6:** SDU Scan Registers (Continued)

| Instruction Decode<br>(TDI->TDO) | Instruction Names | Register Name                                          | Width |
|----------------------------------|-------------------|--------------------------------------------------------|-------|
| 10101                            | SDU_STAT          | SDU Status Register                                    | 32    |
| 01101                            | SDU_MACCTL        | Memory Access Control Register                         | 5     |
| 11101                            | SDU_MACADDR       | Memory Access Address Register                         | 32    |
| 00011                            | SDU_MACDATA       | Memory Access Data Register                            | 32    |
| 10011                            | SDU_DMARD         | DMA Read Data Register                                 | 32    |
| 01011                            | SDU_DMAWD         | DMA Write Data Register                                | 32    |
| 10111                            | SDU_SCANMODE      | Scan Mode Enable Register for DFT Scan Chains          | 1     |
| 11011                            | SDU_SCANATSPEED   | Scan At Speed Mode Enable Register for DFT Scan Chains | 1     |

## Memory Access Controller (MAC)

The MAC implements control functions to provide support for access to system memory space (for example L1, L2, L3, System MMRs). These functions support debug host features like Background Telemetry Channel (BTC), Code and Data download. The MAC supports two types of functions

1. MAC Direct access
2. MAC DMA access

### MAC Direct Access

The MAC provides direct access to system memory and system MMRs. This function provides access to individual addresses through address (MACADDR), data (MACDATA), and control (MACCTL) registers. When

the debug host needs to perform a memory access it loads these three registers with address, data, and control values. The act of writing to the control register triggers the MAC state machine to initiate a transfer request on the MAC system interface.

The `MACRDY` bit in the `SDU_STAT` register indicates when the MAC is ready for a new transfer. In the case of the previous operation being a read transfer, `MACRDY` also indicates that read data has been returned in the `MACDATA` register.

## **MAC DMA Access**

The MAC provides DMA access to system memory to improve the efficiency of debug host communication. The MAC DMA interface provides the following:

- DMA FIFO
- DMA data registers
  - DMA Write Data register (`SDU_DMAWD`)
  - DMA Read Data register (`SDU_DMARD`)

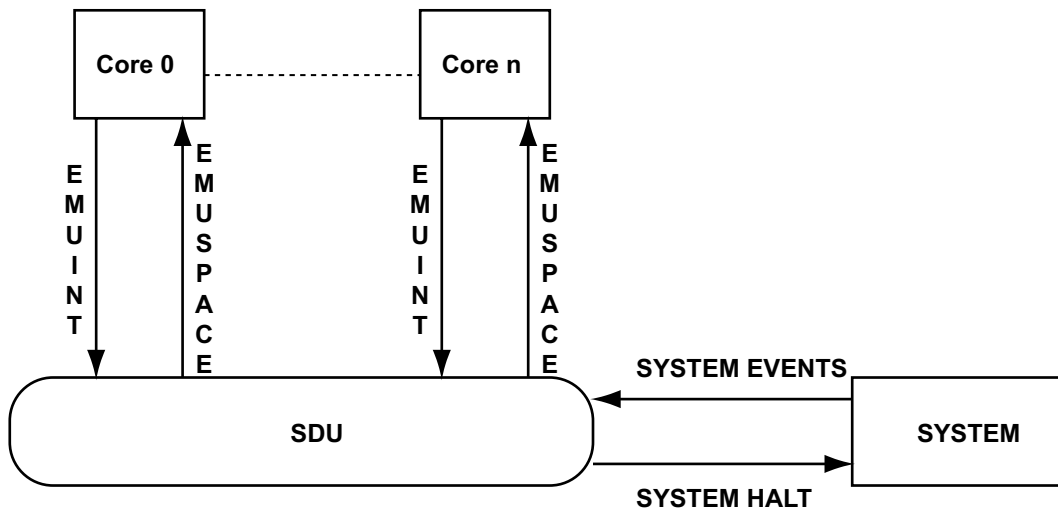
The DMA FIFO is 4-bytes wide and connected to a system DMA channel on one side and to the `SDU_DMARD` and `SDU_DMAWD` registers on the other.

The DMA Interface registers act as the interface points to the DMA FIFO. Depending upon the direction of a DMA transfer, the `SDU_DMAWD` is used to fill the DMA FIFO, or the `SDU_DMARD` is used to empty the DMA FIFO.

## **Group Halt**

The SDU provides support for a user defined group halt model. The model allows for the masters and slaves of the group halt assertion to be selectable. Each master provides support for event signaling to the SDU for group halt assertion. Each slave provides support for responding to a group halt assertion.

The programming model consists primarily of a system MMR in the SDU (`SDU_GHLT`) to configure the masters and slaves of group halt. An overview of the signaling involved for group halt is shown in the following figure (example signal names shown for illustration purpose only):



## Group Halt Status

Group Halt status and Group Halt Cause (master) are captured in the `SDU_STAT` register. The cause is captured upon assertion of the status. First cause is the only one captured. This information will be used to indicate to the debug host that group halt was asserted and which master generated the group halt. The System Halt signal (from the Group Halt Overview figure) is the indication for non-core system masters to halt if configured to do so. `SDU_STAT.GHLT` is the source of the System Halt if System is selected as a slave. For the system to resume (deassertion of System Halt), `SDU_STAT.GHLT` must be cleared by a W1C operation.

## Group Halt Masters

If “core n” generates a debug event (for example software breakpoint) it asserts its debug mode indication signal (for example `EMUSPACE`) to the SDU. If “core n” is enabled as a master of group halt, the SDU sets `GHLT` and capture `GHLTC` in `SDU_STAT`.

If the system generates a debug event (for example SDU trigger input, system watchpoint event) it asserts the corresponding signal to the SDU. If the system is enabled as a master of group halt, the SDU sets `GHLT` and captures `GHLTC` in the `SDU_STAT` register.

System debug events may be disabled at the source as necessary (for example System Watchpoint event indication to SDU may be disabled in the System Watchpoint Unit).

## Group Halt Slaves

When Group Halt status (`SDU_STAT.GHLT`) is asserted, the SDU asserts the debug request signal (for example `EMUINT`) to selected cores according to the settings in the `SDU_GHLT` register. The debug request to a core is asserted based on the transition of `GHLT` from 0 to 1).

When Group Halt status (`SDU_STAT.GHLT`) is asserted, the SDU asserts the System Halt signal if System is selected as a slave in the `SDU_GHLT` register.

Each system master has a dedicated control bit to specify its response to a System Halt signal assertion. Typically this response is to halt the system which may have the following local implications:

1. Stop issuing system transactions
2. Pause the entire block
3. Both

## **SDU Programming Concepts**

### **Core Control**

Cores are directly controlled through emulation space. Core emulation space is entered by generating the core emulation interrupt. To do this, emulation and the emulation interrupt must be enabled in the appropriate core control register.

The SDU generates an emulator interrupt with the rising edge of the TMS pin when the JTAG TAP controller in SDU is in the Run-Test-Idle state. The processor core(s) enters emulation space in response to the emulation interrupt from SDU. The SDU generates an emulation interrupt to a core if the specific core is selected in the SDU control register (`EHLT` field). Once in emulation space, the core executes operations as directed by the debug host. Processor instructions are scanned in and executed using the core emulation instruction register (for example `EMUIR`) and the core emulation data register (for example `EMUDAT`). The JTAG TAP controller is used to perform the scan operations.

### **Memory and Register Access**

Memory and register access is performed using one of the following two methods.

- Direct: Memory and registers that are mapped into the system address space may be accessed directly by the SDU system interfaces (for example `MAC`).
- Indirect: Memory and registers that are not mapped into the system address space may only be accessed indirectly through core instruction execution while in emulation space (see Core Control).

### **Background Telemetry Channel (BTC)**

The BTC provides communication between the system and the debug host. System memory is required for BTC operation and must be allocated by the application for BTC use. The processor application

sends/receives messages to/from the debug host through the dedicated system memory space. The debug host accesses the BTC system memory space through the SDU MAC (the DMA channel).

## Statistical Profiling

Cores may support runtime statistical profiling by providing a register that samples the core PC in a pseudo-random fashion. The debug host accesses the core register (for example `EMUPC`) through JTAG scan operations.

## Power Management Support

A processor may support various power saving modes which may impair the ability for a debug host to communicate with the processor resources because the debug host may not be able to detect when the processor enters or exits such modes. The SDU provides the following support for the debug host to determine the power mode.

- **Core Clock(s) Stopped:** Power management may support a mode (for example Sleep) which stops the core clock(s). In this mode the SDU is unable to communicate with some system resources (for example core registers, core memory). The power management resource (for example Dynamic Power Management or DPM) typically remains accessible. The debug host, through the SDU MAC, may access the status register(s) in DPM to determine the state of the processor.
- **System Clock(s) Stopped:** Power Management may support a mode (for example Deep Sleep) which stops the system clock(s). In this mode the SDU is unable to communicate with all system resources including the power management resource (for example Dynamic Power Management). The SDU provides an input and associated status bit for the DPM to provide an indication of this power mode. The debug host may scan the SDU status register (`SDU_STAT`) through JTAG.

## System Reset Support

The SDU may indirectly generate a system reset. When set, the system reset bit (`SYSRST`) in the CTL register signals a system reset request to the Reset Control Unit (RCU). In response, the RCU asserts the system reset signal. Other events in the system can also cause the RCU to issue a system reset. In response to the system reset signal from RCU, all registers in the SDU are reset, except for the CTL register. The SDU provides an indication that system reset has occurred (`SDU_STAT.SYSRST`). See registers description section for more detail.

## SDU Register Descriptions

System Debug Unit (SDU) contains the following registers.

**Table 35-7:** SDU Module Registers

| Name        | Description                    |
|-------------|--------------------------------|
| SDU_IDCODE  | ID Code Register               |
| SDU_CTL     | Control Register               |
| SDU_STAT    | Status Register                |
| SDU_MACCTL  | Memory Access Control Register |
| SDU_MACADDR | Memory Access Address Register |
| SDU_MACDATA | Memory Access Data Register    |
| SDU_DMARD   | DMA Read Data Register         |
| SDU_DMAWD   | DMA Write Data Register        |
| SDU_MSG     | Message Register               |
| SDU_MSG_SET | Message Set Register           |
| SDU_MSG_CLR | Message Clear Register         |
| SDU_GHLT    | Group Halt Register            |

## ID Code Register

The SDU ID code register (SDU\_IDCODE) contains bit fields describing the processor silicon revision, product identification, and manufacturer identification.



SDU\_IDCODE: ID Code Register - R/NW

Reset = 0x0000 00cb

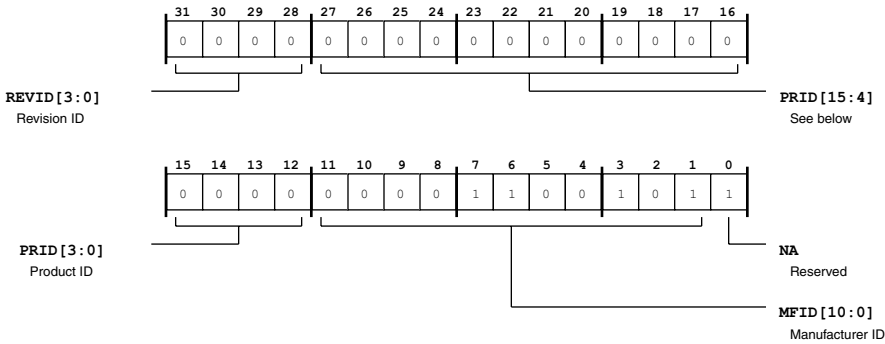


Figure 35-2: SDU\_IDCODE Register Diagram

Table 35-8: SDU\_IDCODE Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:28   | REVID    | Revision ID.            |
| 27:12   | PRID     | Product ID.             |
| 11:1    | MFID     | Manufacturer ID.        |
| 0       | NA       | Reserved.               |

Control Register

The SDU control register (SDU\_CTL) contains bit fields enabling or disabling emulator features. This register may be reset only by asserting the SDU\_TRST pin.

SDU\_CTL: Control Register - R/NW

Reset = 0x0000 0000

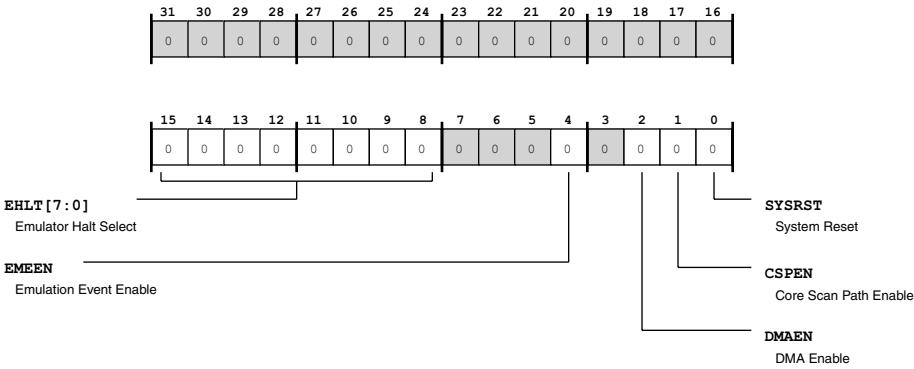


**Figure 35-3:** SDU\_CTL Register Diagram

**Table 35-9:** SDU\_CTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                       |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 15:8    | EHLT     | Emulator Halt Select.<br>The SDU_CTL . EHLT selects system components to halt on assertion of the SDU_STAT . EME bit.                                                                                                                                                                         |
| 4       | EMEEN    | Emulation Event Enable.<br>The SDU_CTL . EMEEN enables the assertion of the SDU_STAT . EME bit on the rising edge of the SDU_TMS pin input.                                                                                                                                                   |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |
| 2       | DMAEN    | DMA Enable.<br>The SDU_CTL . DMAEN enables the operation of the SDU_DMARD and SDU_DMAWD registers. When the SDU_CTL . DMAEN bit is cleared, the SDU_DMARD and SDU_DMAWD registers are disabled, and the SDU_STAT . DMARDRDY and SDU_STAT . DMAWDRDY bit (associated status bits) are cleared. |
|         |          | 0      Disable                                                                                                                                                                                                                                                                                |
|         |          | 1      Enable                                                                                                                                                                                                                                                                                 |

**Table 35-9:** SDU\_CTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                       |                                                                                                                                                          |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | CSPEN    | Core Scan Path Enable.<br>The <code>SDU_CTL.CSPEN</code> enables the JTAG TAP controllers, adding them to the JTAG scan path. |                                                                                                                                                          |
|         |          | 0                                                                                                                             | Disable<br>Signals the (RCU) module to assert system reset, resetting all resources in the SDU SCLK domain except for the <code>SDU_CTL</code> register. |
|         |          | 1                                                                                                                             | Enable                                                                                                                                                   |
| 0       | SYSRST   | System Reset.<br>The <code>SDU_CTL.SYSRST</code> forces a hardware reset when set.                                            |                                                                                                                                                          |
|         |          | 0                                                                                                                             | Inactive                                                                                                                                                 |
|         |          | 1                                                                                                                             | Active                                                                                                                                                   |

## Status Register

The SDU status register (`SDU_STAT`) contains bit fields describing debug-related processor status.

## SDU\_STAT: Status Register - R/W

Reset = 0x0000 0101

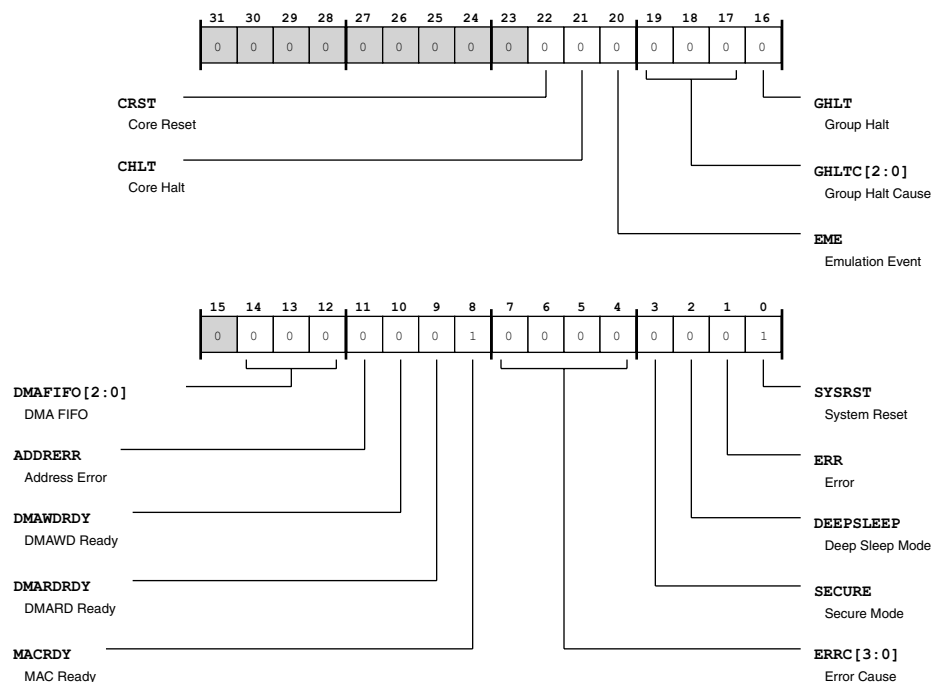


**Figure 35-4: SDU\_STAT Register Diagram**

**Table 35-10:** SDU\_STAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                           |          |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 22      | CRST     | Core Reset.<br>The <code>SDU_STAT.CRST</code> indicates the core reset has been asserted to one or more of the cores in the system. The core debug status register indicates which core or cores have been reset. |          |
|         |          | 0                                                                                                                                                                                                                 | Inactive |
|         |          | 1                                                                                                                                                                                                                 | Active   |
| 21      | CHLT     | Core Halt.<br>The <code>SDU_STAT.CHLT</code> indicates one or more of the cores in the system are halted. The core debug status register indicates which core or cores have halted.                               |          |
|         |          | 0                                                                                                                                                                                                                 | Inactive |
|         |          | 1                                                                                                                                                                                                                 | Active   |

**Table 35-10:** SDU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |          |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 20      | EME      | Emulation Event.<br>The <code>SDU_STAT.EME</code> indicates an emulation event is active and is set when the <code>SDU_TMS</code> pin transitions from 0-to-1 while the <code>SDU_CTL.EMEEN</code> bit is set. The <code>SDU_STAT.EME</code> bit transition to 1 serves as the source for core halt assertion as part of <code>SDU_CTL.EMEEN</code> selection. The emulator clears <code>SDU_STAT.EME</code> with a W1C operation, clearing <code>SDU_STAT.EME</code> for the next core halt by emulation. <code>SDU_STAT.EME</code> is the source for system halt assertion as part of <code>SDU_CTL.EHLT</code> selection. So, event response software must clear <code>SDU_STAT.EME</code> to resume system operation. |          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Inactive |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Active   |
| 19:17   | GHLTC    | Group Halt Cause.<br>The <code>SDU_STAT.GHLTC</code> identifies the group halt master. The SDU updates the <code>SDU_STAT.GHLTC</code> bits on <code>SDU_STAT.GHLT</code> assertion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | System   |
|         |          | ...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Reserved |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Core 0   |
|         |          | 2                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Core 1   |
|         |          | 3                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Reserved |
|         |          | 7                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Reserved |
| 16      | GHLT     | Group Halt.<br>The <code>SDU_STAT.GHLT</code> indicates that group halt has been asserted by one of the group halt masters. On <code>SDU_STAT.GHLT</code> assertion, the SDU updates <code>SDU_STAT.GHLTC</code> with the value identifying the group halt cause. <code>SDU_STAT.GHLT</code> is the source for system halt assertion as part of group halt slave selection with the <code>SDU_GHLT.SS0-SDU_GHLT.SS2</code> bits. So, halt response software must clear <code>SDU_STAT.GHLT</code> to resume system operation.                                                                                                                                                                                             |          |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Inactive |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | Active   |

**Table 35-10:** SDU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 14:12   | DMAFIFO  | DMA FIFO.<br>The <code>SDU_STAT.DMAFIFO</code> indicates DMA FIFO full/empty status.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 0 Empty                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Empty < FIFO <= 1/4 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 2 1/4 Full < FIFO <= 1/2 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 3 1/2 Full < FIFO <= 3/4 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|         |          | 4 3/4 Full < FIFO < Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|         |          | 5 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 6 Reserved                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 7 Full                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 11      | ADDRERR  | Address Error.<br>The <code>SDU_STAT.ADDRERR</code> indicates an attempted write to a read-only register or an access an invalid address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 0 Inactive                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|         |          | 1 Active                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 10      | DMAWDRDY | DMAWD Ready.<br>The <code>SDU_STAT.DMAWDRDY</code> indicates whether the <code>SDU_DMAWD</code> register is ready for the next transfer. When new data is written to the <code>SDU_DMAWD</code> register, the <code>SDU_STAT.DMAWDRDY</code> bit is cleared. When the write data is pushed into the DMA FIFO from the <code>SDU_DMAWD</code> register, the <code>SDU_STAT.DMAWDRDY</code> bit is set, indicating that it is safe to bring in next piece of write data for the DMA transfer. If <code>SDU_DMAWD</code> is written while <code>SDU_STAT.DMAWDRDY</code> is not set, the <code>SDU_STAT</code> register's <code>SDU_STAT.ERR</code> bit in is set (indicating an overflow condition) and the <code>SDU_STAT.ERRC</code> field is updated (indicating the overflow). |
|         |          | 0 Not Ready                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|         |          | 1 Ready                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

**Table 35-10:** SDU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 9       | DMARDRDY | DMARD Ready.<br>The <code>SDU_STAT.DMARDRDY</code> indicates whether the <code>SDU_DMARD</code> register is ready for the next transfer. When data is read from the <code>SDU_DMARD</code> register, the <code>SDU_STAT.DMARDRDY</code> bit is cleared. When the new read data is loaded into the <code>SDU_DMARD</code> register from the DMA FIFO, the <code>SDU_STAT.DMARDRDY</code> bit is set, indicating that it is safe read (scan out) the next piece of read data for the DMA transfer. If <code>SDU_DMARD</code> is read (scanned out) while <code>SDU_STAT.DMARDRDY</code> is not set, the <code>SDU_STAT</code> register's <code>SDU_STAT.ERR</code> bit in is set (indicating an underflow condition) and the <code>SDU_STAT.ERRC</code> field is updated (indicating the underflow). The <code>SDU_STAT.DMARDRDY</code> bit is cleared when the read data is placed on the scan chain during JTAG scan operation. |           |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Not Ready |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Ready     |
| 8       | MACRDY   | MAC Ready.<br>The <code>SDU_STAT.MACRDY</code> indicates whether the <code>SDU_MACADDR</code> and <code>SDU_MACDATA</code> registers are ready for the next transfer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |           |
|         |          | 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Not Ready |
|         |          | 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Ready     |

**Table 35-10:** SDU\_STAT Register Fields (Continued)

| Bit No. | Bit Name  | Description/Enumeration                                                                                                                                                                                                         |
|---------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 7:4     | ERRC      | <p><b>Error Cause.</b><br/>The <code>SDU_STAT.ERRC</code> identifies the error type. The SDU updates <code>SDU_STAT.ERRC</code> on assertion of the <code>SDU_STAT.ERR</code> bit.</p>                                          |
|         |           | ... Reserved                                                                                                                                                                                                                    |
|         |           | 0 MAC over/underflow                                                                                                                                                                                                            |
|         |           | 1 DMARD underflow                                                                                                                                                                                                               |
|         |           | 2 DMAWD overflow                                                                                                                                                                                                                |
|         |           | 3 DMA Error                                                                                                                                                                                                                     |
|         |           | 4 MAC Bus Error                                                                                                                                                                                                                 |
|         |           | 5 Reserved                                                                                                                                                                                                                      |
|         |           | 15 Reserved                                                                                                                                                                                                                     |
| 3       | SECURE    | <p><b>Secure Mode.</b><br/>The <code>SDU_STAT.SECURE</code> indicates whether the processor is operating in secure mode.</p>                                                                                                    |
|         |           | 0 Inactive                                                                                                                                                                                                                      |
|         |           | 1 Active                                                                                                                                                                                                                        |
| 2       | DEEPSLEEP | <p><b>Deep Sleep Mode.</b><br/>The <code>SDU_STAT.DEEPSLEEP</code> indicates whether the processor is operating in DEEPSLEEP power mode. The processor's transition to DEEPSLEEP mode sets <code>SDU_STAT.DEEPSLEEP</code>.</p> |
|         |           | 0 Inactive                                                                                                                                                                                                                      |
|         |           | 1 Active                                                                                                                                                                                                                        |



Table 35-10: SDU\_STAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                               |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | ERR      | Error.<br>The SDU_STAT.ERR indicates an SDU related error. The SDU updates SDU_STAT.ERRC on assertion SDU_STAT.ERR with the value identifying the error cause. SDU_STAT.ERRC only captures the first error on SDU_STAT.ERR assertion. |
|         |          | 0 Inactive                                                                                                                                                                                                                            |
|         |          | 1 Active                                                                                                                                                                                                                              |
| 0       | SYSRST   | System Reset.<br>The SDU_STAT.SYSRST indicates that system reset has been asserted. The assertion of system reset sets SDU_STAT.SYSRST.                                                                                               |
|         |          | 0 Inactive                                                                                                                                                                                                                            |
|         |          | 1 Active                                                                                                                                                                                                                              |

Memory Access Control Register

The SDU memory access control register (SDU\_MACCTL) contains bit fields configuring memory access features.

SDU\_MACCTL: Memory Access Control Register - R/W

Reset = 0x0000 0002

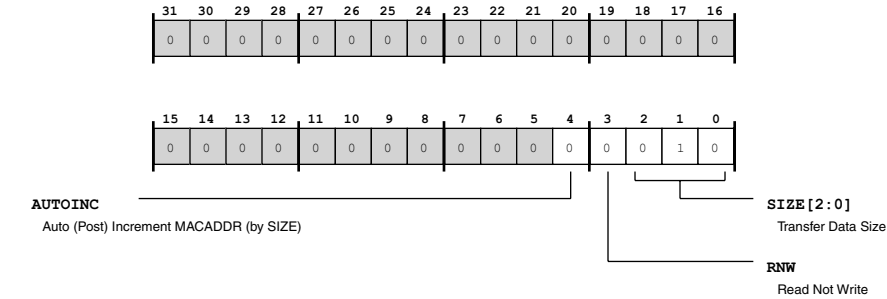


Figure 35-5: SDU\_MACCTL Register Diagram

**Table 35-11:** SDU\_MACCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                     |
|---------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4       | AUTOINC  | Auto (Post) Increment MACADDR (by SIZE).<br>The <code>SDU_MACCTL.AUTOINC</code> enables auto (post) increment of the <code>SDU_MACADDR</code> register value by the <code>SDU_MACCTL.SIZE</code> selection. |
|         |          | 0      Inactive                                                                                                                                                                                             |
|         |          | 1      Active                                                                                                                                                                                               |
| 3       | RNW      | Read Not Write.<br>The <code>SDU_MACCTL.RNW</code> selects read or write operation for the MAC transfer.                                                                                                    |
|         |          | 0      Write                                                                                                                                                                                                |
|         |          | 1      Read                                                                                                                                                                                                 |
| 2:0     | SIZE     | Transfer Data Size.<br>The <code>SDU_MACCTL.SIZE</code> selects the size of the data in the <code>SDU_MACDATA</code> register for the MAC transfer.                                                         |
|         |          | 0      8b                                                                                                                                                                                                   |
|         |          | ...      Reserved                                                                                                                                                                                           |
|         |          | 1      16b                                                                                                                                                                                                  |
|         |          | 2      32b                                                                                                                                                                                                  |
|         |          | 3      Reserved                                                                                                                                                                                             |
|         |          | 7      Reserved                                                                                                                                                                                             |

## Memory Access Address Register

The SDU memory access address register (`SDU_MACADDR`) contains the address of the MAC transfer.

SDU\_MACADDR: Memory Access Address Register - R/W

Reset = 0x0000 0000

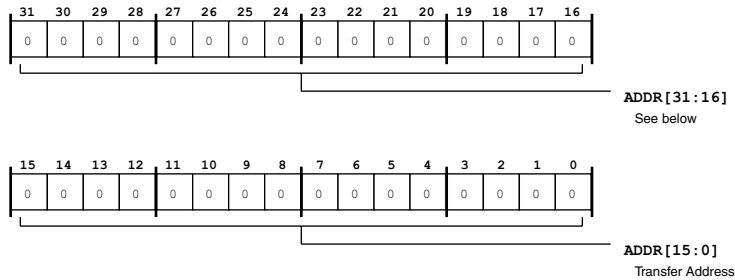


Figure 35-6: SDU\_MACADDR Register Diagram

Table 35-12: SDU\_MACADDR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                           |
|---------|----------|-----------------------------------------------------------------------------------|
| 31:0    | ADDR     | Transfer Address.<br>The SDU_MACADDR.ADDR holds the address for the MAC transfer. |

Memory Access Data Register

The SDU memory access data register (SDU\_MACDATA) contains the data for the MAC transfer.

SDU\_MACDATA: Memory Access Data Register - R/W

Reset = 0x0000 0000

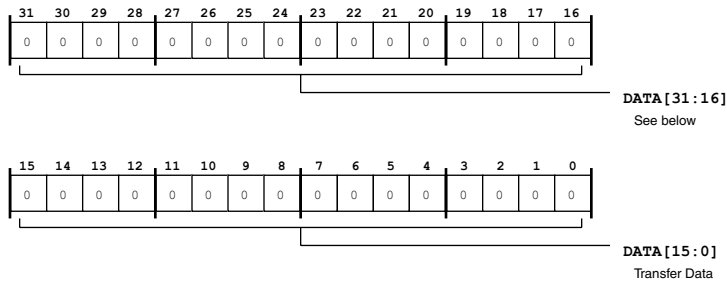


Figure 35-7: SDU\_MACDATA Register Diagram

Table 35-13: SDU\_MACDATA Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                     |
|---------|----------|-----------------------------------------------------------------------------|
| 31:0    | DATA     | Transfer Data.<br>The SDU_MACDATA.DATA holds the data for the MAC transfer. |

## DMA Read Data Register

The SDU DMA read data register (SDU\_DMARD) contains the data for the DMA read transfer.

**SDU\_DMARD: DMA Read Data Register - R/W**

Reset = 0x0000 0000

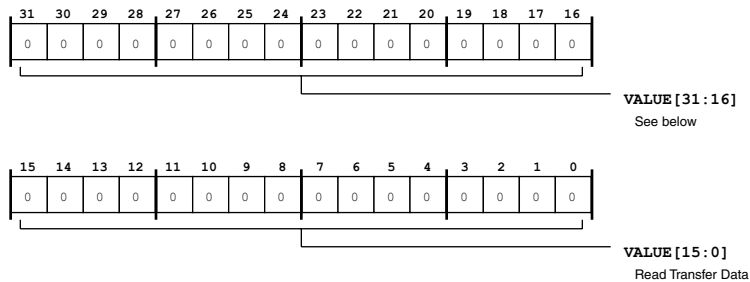


**Figure 35-8:** SDU\_DMARD Register Diagram

**Table 35-14:** SDU\_DMARD Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                              |
|---------|----------|--------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Read Transfer Data.<br>The SDU_DMARD.VALUE holds the data for the DMA read transfer. |

## DMA Write Data Register

The SDU DMA write data register (SDU\_DMAWD) contains the data for the DMA write transfer.

**SDU\_DMAWD: DMA Write Data Register - R/W**

Reset = 0x0000 0000

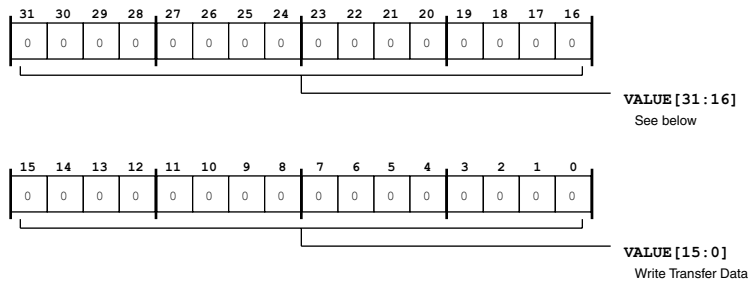


**Figure 35-9:** SDU\_DMAWD Register Diagram

Table 35-15: SDU\_DMAWD Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                |
|---------|----------|----------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Write Transfer Data.<br>The SDU_DMAWD.VALUE holds the data for the DMA write transfer. |

Message Register

The SDU message register (SDU\_MSG) holds user defined messages to pass to or to get from the debugger.

SDU\_MSG: Message Register - R/W

Reset = 0x0000 0000

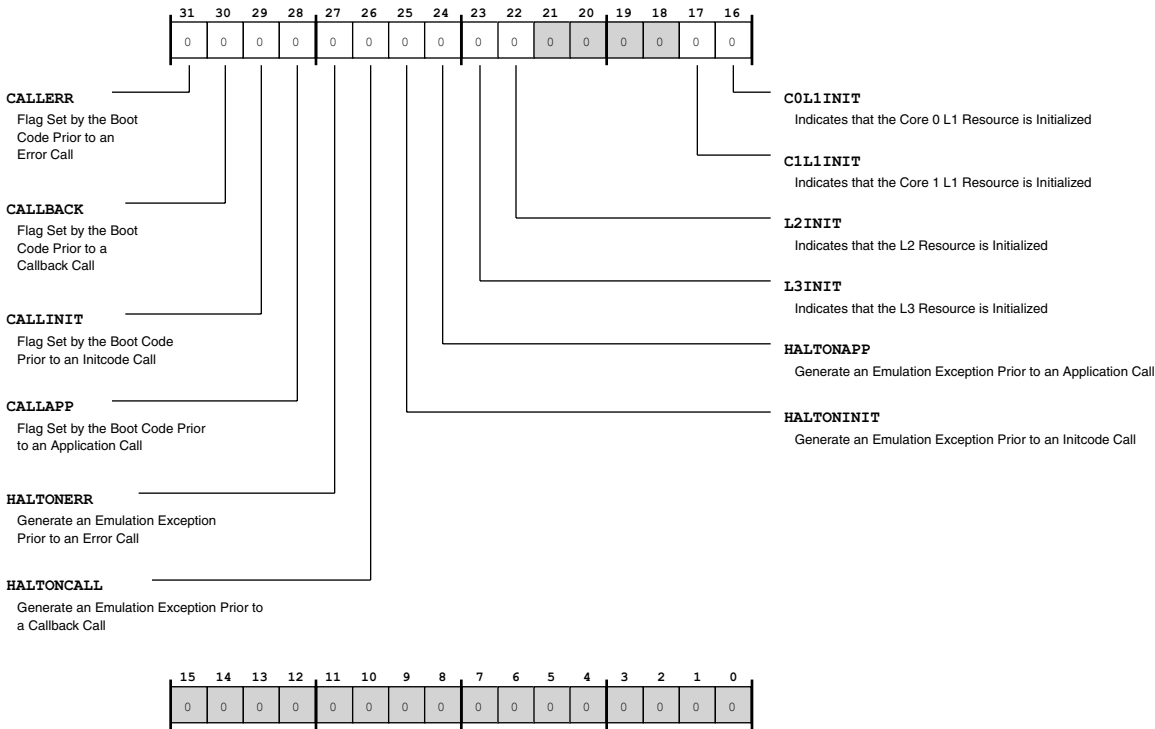


Figure 35-10: SDU\_MSG Register Diagram

**Table 35-16:** SDU\_MSG Register Fields

| Bit No. | Bit Name   | Description/Enumeration                                       |
|---------|------------|---------------------------------------------------------------|
| 31      | CALLERR    | Flag Set by the Boot Code Prior to an Error Call.             |
| 30      | CALLBACK   | Flag Set by the Boot Code Prior to a Callback Call.           |
| 29      | CALLINIT   | Flag Set by the Boot Code Prior to an Initcode Call.          |
| 28      | CALLAPP    | Flag Set by the Boot Code Prior to an Application Call.       |
| 27      | HALTONERR  | Generate an Emulation Exception Prior to an Error Call.       |
| 26      | HALTONCALL | Generate an Emulation Exception Prior to a Callback Call.     |
| 25      | HALTONINIT | Generate an Emulation Exception Prior to an Initcode Call.    |
| 24      | HALTONAPP  | Generate an Emulation Exception Prior to an Application Call. |
| 23      | L3INIT     | Indicates that the L3 Resource is Initialized.                |
| 22      | L2INIT     | Indicates that the L2 Resource is Initialized.                |
| 17      | C1L1INIT   | Indicates that the Core 1 L1 Resource is Initialized.         |
| 16      | C0L1INIT   | Indicates that the Core 0 L1 Resource is Initialized.         |

## Message Set Register

The SDU message set register (SDU\_MSG\_SET) is a write-1-set alias for the SDU\_MSG register.

### SDU\_MSG\_SET: Message Set Register - R/W

Reset = 0x0000 0000

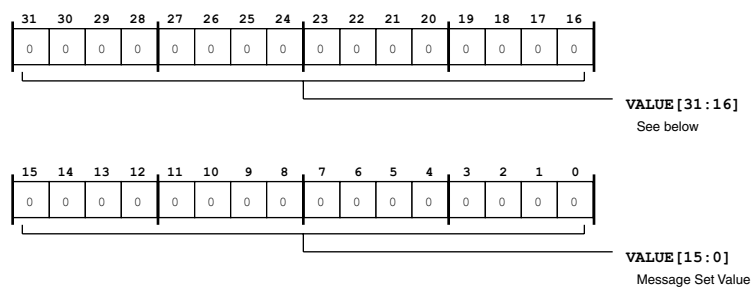


**Figure 35-11:** SDU\_MSG\_SET Register Diagram

**Table 35-17:** SDU\_MSG\_SET Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                        |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Message Set Value.<br>The SDU_MSG_SET.VALUE acts as a write-1-set alias for the SDU_MSG register. Writing a 1 to any bit in SDU_MSG_SET sets the corresponding bit in SDU_MSG. |

Message Clear Register

The SDU message clear register (SDU\_MSG\_CLR) is a write-1-clear alias for the SDU\_MSG register.

SDU\_MSG\_CLR: Message Clear Register - R/W

Reset = 0x0000 0000

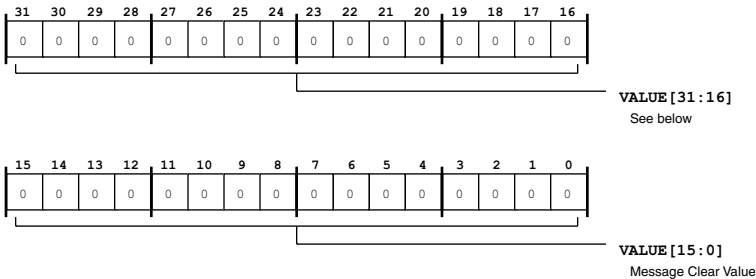


**Figure 35-12:** SDU\_MSG\_CLR Register Diagram

**Table 35-18:** SDU\_MSG\_CLR Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                              |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:0    | VALUE    | Message Clear Value.<br>The SDU_MSG_CLR.VALUE acts as a write-1-clear alias for the SDU_MSG register. Writing a 1 to any bit in SDU_MSG_CLR clears the corresponding bit in SDU_MSG. |

Group Halt Register

The SDU group halt register (SDU\_GHLT) configures the master and slave selections for group halt.

SDU\_GHLT: Group Halt Register - R/W

Reset = 0x0000 0000

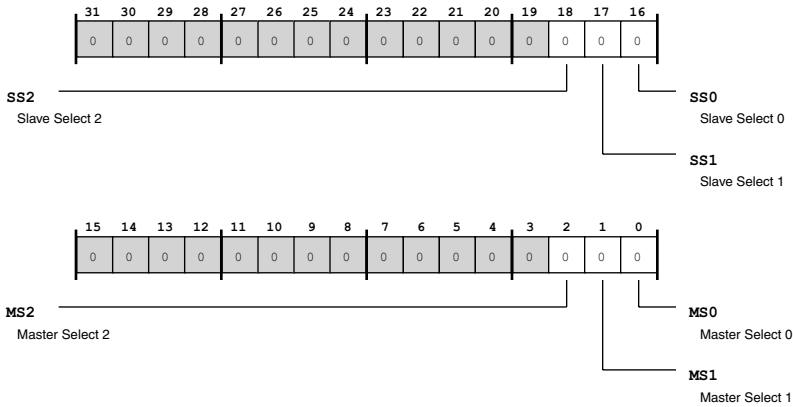


Figure 35-13: SDU\_GHLT Register Diagram

Table 35-19: SDU\_GHLT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                     |              |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| 18      | SS2      | Slave Select 2.<br>The SDU_GHLT.SS2 selects core 1 slave for group halt. On SDU_STAT.GHLT assertion, a halt command is issued to the slave. |              |
|         |          | 0                                                                                                                                           | Not Selected |
|         |          | 1                                                                                                                                           | Selected     |
| 17      | SS1      | Slave Select 1.<br>The SDU_GHLT.SS1 selects core 0 slave for group halt. On SDU_STAT.GHLT assertion, a halt command is issued to the slave. |              |
|         |          | 0                                                                                                                                           | Not Selected |
|         |          | 1                                                                                                                                           | Selected     |
| 16      | SS0      | Slave Select 0.<br>The SDU_GHLT.SS0 selects system slave for group halt. On SDU_STAT.GHLT assertion, a halt command is issued to the slave. |              |
|         |          | 0                                                                                                                                           | Not Selected |
|         |          | 1                                                                                                                                           | Selected     |



**Table 35-19:** SDU\_GHLT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                   |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 2       | MS2      | Master Select 2.<br>The <code>SDU_GHLT.MS2</code> selects core 1 master as a source for group halt ( <code>SDU_STAT.GHLT</code> ) assertion.              |
|         |          | 0      Not Selected                                                                                                                                       |
|         |          | 1      Selected                                                                                                                                           |
| 1       | MS1      | Master Select 1.<br>The <code>SDU_GHLT.MS1</code> selects core 0 master as a source for group halt ( <code>SDU_STAT.GHLT</code> ) assertion.              |
|         |          | 0      Not Selected                                                                                                                                       |
|         |          | 1      Selected                                                                                                                                           |
| 0       | MS0      | Master Select 0.<br>The <code>SDU_GHLT.MS0</code> selects system debug events master as a source for group halt ( <code>SDU_STAT.GHLT</code> ) assertion. |
|         |          | 0      Not Selected                                                                                                                                       |
|         |          | 1      Selected                                                                                                                                           |



## 36 System Watchpoint Unit (SWU)

The system watchpoint unit (SWU) is a single module used for transaction monitoring. The SWU is attached to each system slave through the system crossbar interface and provides ports for all system crossbar address channel signals. The SWU does not have ports for the read/write data channel signals or the low-power interface signals.

Each SWU contains four match groups of registers with associated hardware. These four SWU match groups operate independently, but share common event (interrupt and trigger) outputs. Each match group can monitor either the write or read address channel and can operate in either watchpoint mode or bandwidth mode.

### SWU Features

- Four independent match groups for each SWU
- Each match group can operate in either bandwidth mode or watchpoint mode

### SWU Functional Description

This section describes the function of the SWU match block, interface block and MMR block.

#### Match Block

There are four match blocks for each SWU. Each SWU match block can monitor either the read or write address channel, selected by the `SWU_CTL.DIR` bit, and can operate in either watchpoint or bandwidth mode, selected by the `SWU_CTL.BWEN` bit.

In either mode, the SWU match block can be programmed to match based on address (exact, inclusive/exclusive range), ID (with masking), security, and lock type. All enabled matches are AND'ed together to determine a match.

### SWU Register List

The system watchpoint unit (SWU) provides debug and development support through flexible transaction level and bandwidth monitoring and associated event triggering. The SWU generates an events based on monitoring transactions at the system slaves using four watchpoint match groups. The transaction monitoring within each match group may be filtered by address, ID, direction, and other watchpoint attributes.

An event may be a trace message, trigger, or interrupt. The desired event behavior is programmed into the appropriate system watchpoint controls and attributes. The SWU also provides watchpoint event status reporting, a global lock, and processor reset. A set of registers govern SWU operations. For more information on SWU functionality, see the SWU register descriptions.

**Table 36-1:** SWU Module Registers

| Name      | Description                  |
|-----------|------------------------------|
| SWU_GCTL  | Global Control Register      |
| SWU_GSTAT | Global Status Register       |
| SWU_CTLn  | Control Register n           |
| SWU_LAn   | Lower Address Register n     |
| SWU_UAn   | Upper Address Register n     |
| SWU_IDn   | ID Register n                |
| SWU_CNTn  | Count Register n             |
| SWU_TARGn | Target Register n            |
| SWU_HISTn | Bandwidth History Register n |
| SWU_CURn  | Current Register n           |

## ADSP-BF60x SWU Interrupt List

**Table 36-2:** ADSP-BF60x SWU Interrupt List

| Description | ID  | DMA Channel | Sensitivity |
|-------------|-----|-------------|-------------|
| SWU0 Event  | 133 |             | LEVEL       |
| SWU1 Event  | 134 |             | LEVEL       |
| SWU2 Event  | 135 |             | LEVEL       |
| SWU3 Event  | 136 |             | LEVEL       |
| SWU4 Event  | 137 |             | LEVEL       |

**Table 36-2:** ADSP-BF60x SWU Interrupt List (Continued)

| Description | ID  | DMA Channel | Sensitivity |
|-------------|-----|-------------|-------------|
| SWU5 Event  | 138 |             | LEVEL       |
| SWU6 Event  | 139 |             | LEVEL       |

## ADSP-BF60x SWU Trigger List

**Table 36-3:** Masters ADSP-BF60x SWU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
| SWU0 Event  | 80 | PULSE/EDGE  |
| SWU1 Event  | 81 | PULSE/EDGE  |
| SWU2 Event  | 82 | PULSE/EDGE  |
| SWU3 Event  | 83 | PULSE/EDGE  |
| SWU4 Event  | 84 | PULSE/EDGE  |
| SWU5 Event  | 85 | PULSE/EDGE  |
| SWU6 Event  | 86 | PULSE/EDGE  |
|             |    |             |

**Table 36-4:** Slaves ADSP-BF60x SWU Trigger List

| Description | ID | Sensitivity |
|-------------|----|-------------|
| SWU0 Event  | 80 |             |
| SWU1 Event  | 81 |             |
| SWU2 Event  | 82 |             |
| SWU3 Event  | 83 |             |
| SWU4 Event  | 84 |             |

**Table 36-4:** Slaves ADSP-BF60x SWU Trigger List (Continued)

| Description | ID | Sensitivity |
|-------------|----|-------------|
| SWU5 Event  | 85 |             |
| SWU6 Event  | 86 |             |
|             |    |             |

## SWU Definitions

### Watchpoint Mode

Mode in which transactions are recognized on an exact match. Actions can be configured to be taken after a specified number of matches have occurred.

### Bandwidth Mode

Mode in which transactions are recognized and counted inside sampling window.

## SWU Architectural Concepts

The information in this section provides basic module design concepts.

## SWU Flow Diagram

The following diagram shows the logical program flow of the SWU.

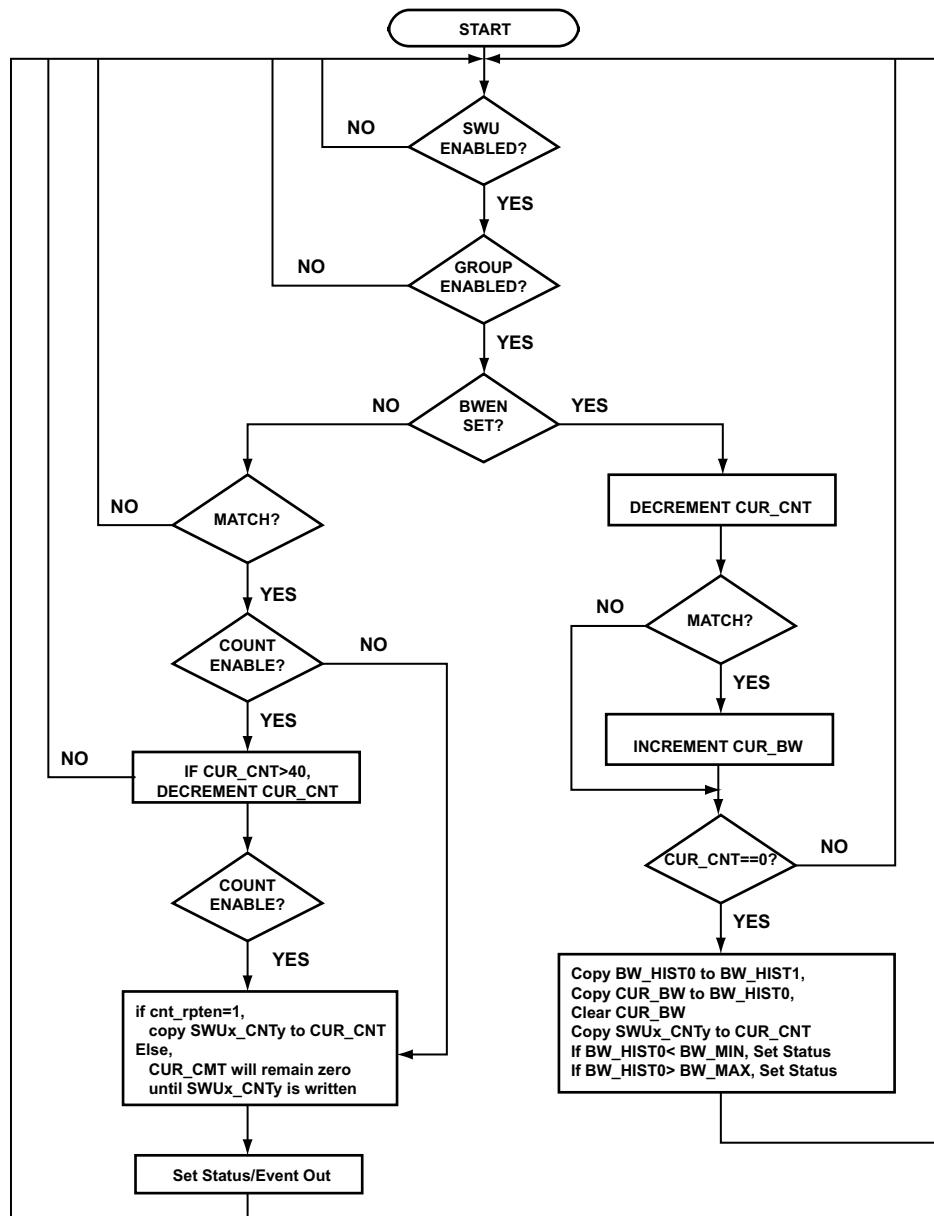


Figure 36-1: SWU Logical Flow

## SCB Interface

The SWU system crossbar interface block latches all transactions on the system crossbar read and write address channels when the SWU\_GCTL register enable bit is set.

## SWU Block Diagram

The following figure shows the SWU block diagram.

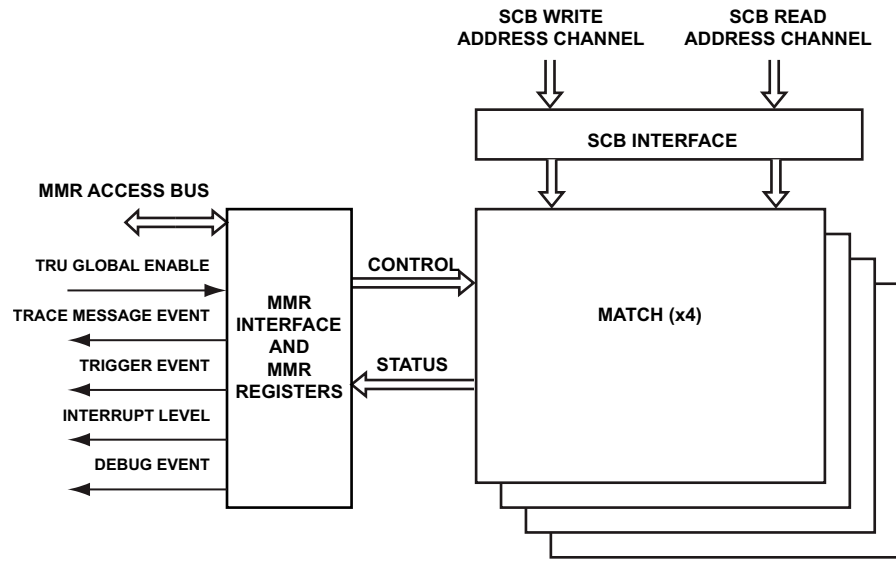


**Figure 36-2:** System Watchpoint Unit Top Level Block Diagram

## SWU Event Control

The SWU can generate the following events when a match occurs and when the event is enabled by configuring the proper bits in the control register.

1. Trace Message
2. Trigger
3. Interrupt
4. Debug

## SWU Interrupts

All interrupts and events from each match block are merged into common outputs.

## SWU Status and Errors

The status and errors are reported in the SWU\_ GSTAT register. The only error that the SWU records is an address error when a write or read attempt is made to the SWU's MMR address space and the register does not exist.

The SWU\_ GSTAT also holds the bits that indicate whether a particular match group sampled transactions that were below a minimum target or above a maximum target in bandwidth mode. There are separate status bits in SWU\_ GSTAT for each match group to indicate whether or not a watchpoint match occurred.



Finally, SWU\_GSTAT also hold status bits to indicate whether or not an interrupt was triggered due to a match event from one of the match groups.

## **SWU Operating Modes**

There are two operating modes supported by the SWU: bandwidth mode and watchpoint mode.

### **Bandwidth Mode**

In bandwidth mode, transactions which match the properties specified in the SWU\_CTLy register are counted during a sampling window determined by the respective SWU\_CNTy register. At the end of the sampling window, results are stored in the SWU\_HISTy register. If the sampled bandwidth falls outside a programmed range, then the programmed action is taken.

### **Watchpoint Mode**

In watchpoint mode, if the CNTEN bit is set, the SWUx\_CURy register is decremented for each match, until it equals zero, at which point any programmed actions are taken. The SWUx\_CURy register is then reloaded from the SWUx\_CNTy register (if CNT\_RPTEN is set), and the cycle repeats. If the CNTEN bit is not set, any programmed actions are taken on every match.

## **SWU Programming Model**

The SWU is used by programming the appropriate registers. Each control register is used to configure various aspects such as the direction of monitoring (reads or writes), whether Bandwidth Mode or Watchpoint Mode is used, setting up which events are triggered when a condition is met while monitoring using the SWU, and other parameters. Supplemental registers such as the lower and upper address boundaries are also configured before enabling.

Once the SWU has been enabled and monitoring conditions are met, events are generated if the SWU was configured to do so.

The global status register can be read to observe the current status of the units.

## **SWU Programming Concepts**

The information for this section is TBD.

## SWU Mode Configuration

The following sections show the steps for configuring the SWU into Bandwidth Mode and Watchpoint Mode.

### Configuring the SWU for Bandwidth Mode

In Bandwidth mode, transactions which match are counted during a sampling window. At the end of the sampling window, results are stored and an action can be taken if the sampled bandwidth goes above and/or falls below a programmed range.

1. Set the SWU\_CTLn.DIR bit depending on whether the SWU should test the match on writes or reads.
2. Set the SWU\_CTLn.ACMPM bits depending on whether address comparisons are made, exact match, matches inside a range or matches outside a range.
3. Optionally, set the SWU\_CTLn.IDCMPEN if ID comparison is desired.
4. Set the SWU\_CTLn.BLENINC bit to increment by burst length or clear it to increment by 1.
5. Set the MAXACT and MINACT bits in the SWU\_CTLn register to enable actions taken when the bandwidth goes above the maximum, or falls below the minimum, respectively.
6. Set the SWU\_CTLn.BWEN=1 to enable Bandwidth mode.
7. Program the lower address register, SWU\_LAn, and upper address register, SWU\_HAn, to define the memory range for comparison.
8. If ID comparison is enabled, program the ID register, SWU\_IDn.
9. Program the count register, SWU\_CNTn, with the number of clock cycles for which the SWU will be counting the number of matches.
10. If the SWU is set to take action when the bandwidth measurement underflows or overflows, the min and max values should be programmed into the SWU\_TARGn register.
11. Enable the SWU

#### *RESULT:*

The SWU counts the number of matches in a pre-defined amount of clock cycles programmed by the user. Lower and upper limits can optionally be defined. If the matches fall outside the limits, an action can be taken.

## Configuring the SWU for Watchpoint Mode

In watchpoint mode, the SWU can trigger a programmed action after every match or after a number of matches. This sequence can be automatically reset.

1. Set the `SWU_CTLn.DIR` bit to test the match on writes or reads.
2. Set the `SWU_CTLn.ACMPM` bits for address comparisons, exact match, matches inside a range or matches outside a range are desired.
3. Optionally set the `SWU_CTLn.IDCMPEN` if ID comparison is desired.
4. Set the `SWU_CTLn.CNTEN` bit to enable the events to be triggered when the count decrements to zero.
5. Optionally set the `SWU_CTLn.CNTRPTEN` bit to automatically reload the counter after it has decremented to zero to start another match sequence.
6. Set the `SWU_CTLn.BWEN = 0` to configure watchpoint mode.
7. Program the lower address register, `SWU_LAn`, and upper address register, `SWU_HAn`, to define the memory range for comparison.
8. If ID comparison is enabled, program the ID register, `SWU_IDn`.
9. Program the count register, `SWU_CNTn`, to determine how many matches occur before the watchpoint group takes action.
10. Enable the SWU.

### RESULT:

The SWU detects and counts down the number of match occurrences. When the counter expires, an action is taken.

## SWU Register Descriptions

System Watchpoint Unit (SWU) contains the following registers.

**Table 36-5:** SWU Module Registers

| Name      | Description             |
|-----------|-------------------------|
| SWU_GCTL  | Global Control Register |
| SWU_GSTAT | Global Status Register  |
| SWU_CTLn  | Control Register n      |

Table 36-5: SWU Module Registers (Continued)

| Name      | Description                  |
|-----------|------------------------------|
| SWU_LAn   | Lower Address Register n     |
| SWU_UAn   | Upper Address Register n     |
| SWU_IDn   | ID Register n                |
| SWU_CNTn  | Count Register n             |
| SWU_TARGn | Target Register n            |
| SWU_HISTn | Bandwidth History Register n |
| SWU_CURn  | Current Register n           |

Global Control Register

The SWU global control register (SWU\_GCTL) provides SWU reset and enable.

SWU\_GCTL: Global Control Register - R/W

Reset = 0x0000 0000

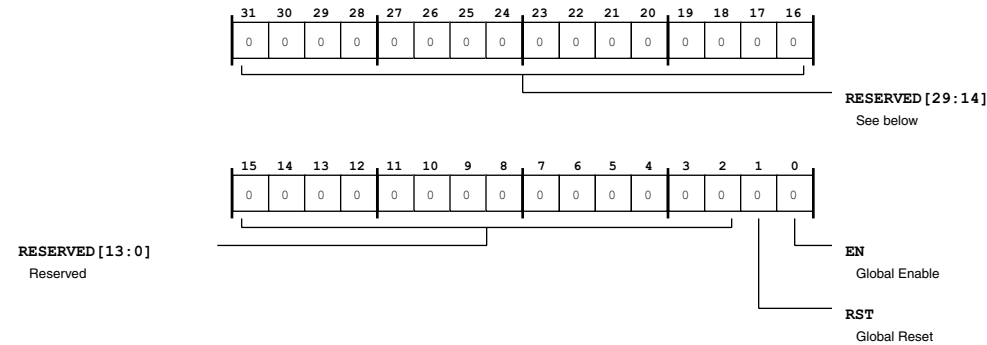


Figure 36-3: SWU\_GCTL Register Diagram

Table 36-6: SWU\_GCTL Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:2    | RESERVED | Reserved.               |

**Table 36-6:** SWU\_GCTL Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                               |           |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 1       | RST      | Global Reset.<br>The SWU_GCTL.RST is write-1-action/read zero and controls the SWU operational state. Setting SWU_GCTL.RST resets all SWU registers to their default values and halts all SWU operations.                                                                                                                             |           |
|         |          | 0                                                                                                                                                                                                                                                                                                                                     | No Action |
|         |          | 1                                                                                                                                                                                                                                                                                                                                     | Reset     |
| 0       | EN       | Global Enable.<br>The SWU_GCTL.EN controls the SWU operational state. Clearing SWU_GCTL.EN halts the execution of all watchpoint and bandwidth tracking operations without resetting status registers or associated signals. Setting SWU_GCTL.EN enables the SWU to begin/resume operation with the current configuration and status. |           |
|         |          | 0                                                                                                                                                                                                                                                                                                                                     | Disable   |
|         |          | 1                                                                                                                                                                                                                                                                                                                                     | Enable    |

## Global Status Register

The SWU global status register (SWU\_GSTAT) contains status bits for all four watchpoint groups.

SWU\_GSTAT: Global Status Register - R/W

Reset = 0x0000 0000

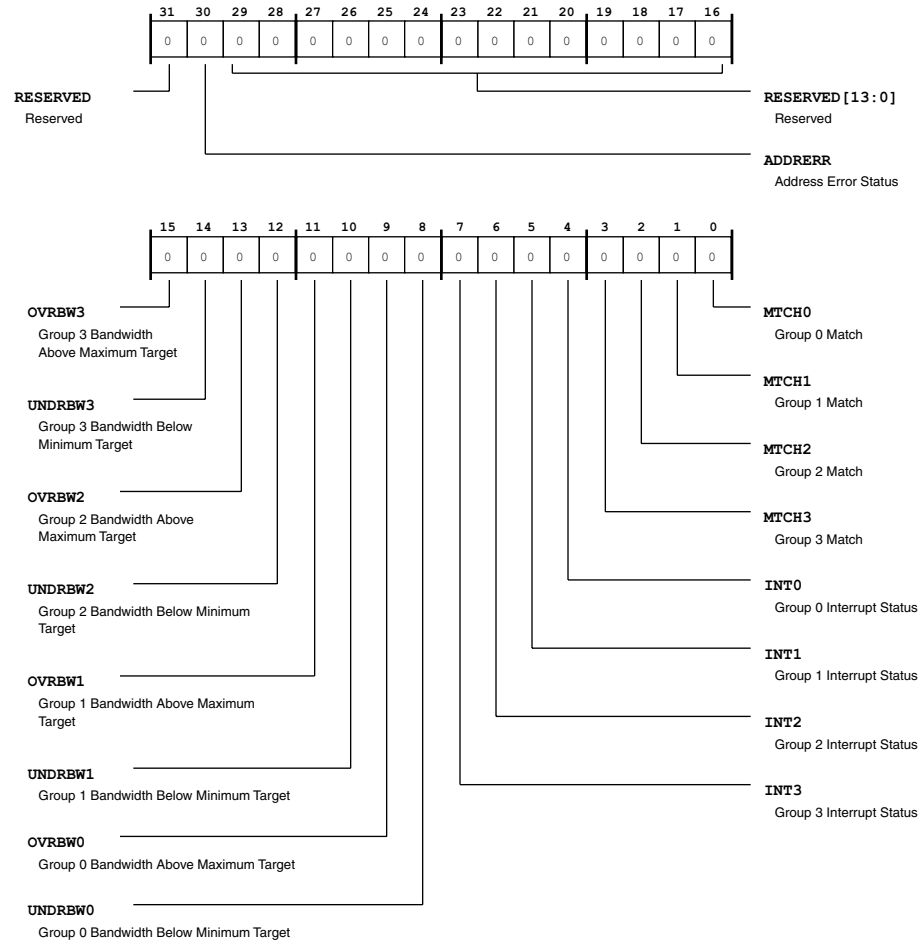


Figure 36-4: SWU\_GSTAT Register Diagram

Table 36-7: SWU\_GSTAT Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                             |          |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| 31      | RESERVED | Reserved.                                                                                                                                           |          |
| 30      | ADDRERR  | Address Error Status.<br>The SWU_GSTAT . ADDRERR indicates that the SWU generated an address error. This status bit is sticky; write-1-to-clear it. |          |
|         |          | 0                                                                                                                                                   | Inactive |
|         |          | 1                                                                                                                                                   | Active   |

**Table 36-7:** SWU\_GSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 29:16   | RESERVED | Reserved.                                                                                                                                                                                                                                                                                                                                                              |
| 15      | OVRBW3   | Group 3 Bandwidth Above Maximum Target.<br>See SWU_GSTAT.OVRBW0 description.                                                                                                                                                                                                                                                                                           |
|         |          | 1 Group 3 was above maximum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 14      | UNDRBW3  | Group 3 Bandwidth Below Minimum Target.<br>See SWU_GSTAT.UNDRBW0 description.                                                                                                                                                                                                                                                                                          |
|         |          | 1 Group 3 was below minimum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 13      | OVRBW2   | Group 2 Bandwidth Above Maximum Target.<br>See SWU_GSTAT.OVRBW0 description.                                                                                                                                                                                                                                                                                           |
|         |          | 1 Group 2 was above maximum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 12      | UNDRBW2  | Group 2 Bandwidth Below Minimum Target.<br>See SWU_GSTAT.UNDRBW0 description.                                                                                                                                                                                                                                                                                          |
|         |          | 1 Group 2 was below minimum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 11      | OVRBW1   | Group 1 Bandwidth Above Maximum Target.<br>See SWU_GSTAT.OVRBW0 description.                                                                                                                                                                                                                                                                                           |
|         |          | 1 Group 1 was above maximum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 10      | UNDRBW1  | Group 1 Bandwidth Below Minimum Target.<br>See SWU_GSTAT.UNDRBW0 description.                                                                                                                                                                                                                                                                                          |
|         |          | 1 Group 1 was below minimum bandwidth                                                                                                                                                                                                                                                                                                                                  |
| 9       | OVRBW0   | Group 0 Bandwidth Above Maximum Target.<br>The SWU_GSTAT.OVRBW0 - SWU_GSTAT.OVRBW3 -- Group 0 through 3 watchpoint bandwidth over maximum target bits. Each maximum bandwidth bit indicate (for each group)s that the measured bandwidth over the period defined by the SWU_CNTn register was over the maximum target. This status bit is sticky; write-1-to-clear it. |
|         |          | 1 Group 0 was above maximum bandwidth                                                                                                                                                                                                                                                                                                                                  |

**Table 36-7:** SWU\_GSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 8       | UNDRBW0  | Group 0 Bandwidth Below Minimum Target.<br>The SWU_GSTAT.UNDRBW0 - SWU_GSTAT.UNDRBW3 -- Group 0 through 3 watchpoint bandwidth below minimum target bits. Each minimum bandwidth bit indicates (for each group) that the measured bandwidth over the period defined by the SWU_CNTn register was below the minimum target. This status bit is sticky; write-1-to-clear it. |
|         |          | 1      Group 0 was below minimum bandwidth                                                                                                                                                                                                                                                                                                                                 |
| 7       | INT3     | Group 3 Interrupt Status.<br>See SWU_GSTAT.INT0 description.                                                                                                                                                                                                                                                                                                               |
|         |          | 0      No Interrupt                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Interrupt Occurred                                                                                                                                                                                                                                                                                                                                                  |
| 6       | INT2     | Group 2 Interrupt Status.<br>See SWU_GSTAT.INT0 description.                                                                                                                                                                                                                                                                                                               |
|         |          | 0      No Interrupt                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Interrupt Occurred                                                                                                                                                                                                                                                                                                                                                  |
| 5       | INT1     | Group 1 Interrupt Status.<br>See SWU_GSTAT.INT0 description.                                                                                                                                                                                                                                                                                                               |
|         |          | 0      No Interrupt                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Interrupt Occurred                                                                                                                                                                                                                                                                                                                                                  |
| 4       | INT0     | Group 0 Interrupt Status.<br>The SWU_GSTAT.INT0 - SWU_GSTAT.INT3 -- Group 0 through 3 interrupt bits. Each interrupt bit indicates (for each group) whether a watchpoint group is contributing to the SWU's interrupt output. This status bit is sticky; write-1-to-clear it.                                                                                              |
|         |          | 0      No interrupt                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1      Interrupt Occurred                                                                                                                                                                                                                                                                                                                                                  |



**Table 36-7:** SWU\_GSTAT Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                            |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 3       | MTCH3    | Group 3 Match.<br>See SWU_GSTAT.MTCH0 description.                                                                                                                                                                                                                                                                                 |
|         |          | 0 No Match                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Group 3 Watchpoint Match                                                                                                                                                                                                                                                                                                         |
| 2       | MTCH2    | Group 2 Match.<br>See SWU_GSTAT.MTCH0 description.                                                                                                                                                                                                                                                                                 |
|         |          | 0 No match                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Group 2 Watchpoint Match                                                                                                                                                                                                                                                                                                         |
| 1       | MTCH1    | Group 1 Match.<br>See SWU_GSTAT.MTCH0 description.                                                                                                                                                                                                                                                                                 |
|         |          | 0 No match                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Group 1 Watchpoint Match                                                                                                                                                                                                                                                                                                         |
| 0       | MTCH0    | Group 0 Match.<br>The SWU_GSTAT.MTCH0 - SWU_GSTAT.MTCH3 -- Group 0 through 3 match bits. Each match bit indicates (for each group) whether a watchpoint match has occurred in a SWU watchpoint group, as controlled by the group's related watchpoint control register (SWU_CTLn). This status bit is sticky; write-1-to-clear it. |
|         |          | 0 No match                                                                                                                                                                                                                                                                                                                         |
|         |          | 1 Group 0 Watchpoint Match                                                                                                                                                                                                                                                                                                         |

## Control Register n

The SWU control registers (SWU\_CTLn) contain watchpoint attribute controls for all four watchpoint groups. These controls include enabling watchpoints, selecting the transaction direction for match, selecting address comparison mode, enabling ID comparison, enabling security comparison, enabling locked comparison, enabling cycle count, enabling count repeat, enabling debug events, enabling interrupts, enabling triggers, enabling trace messages, enabling bandwidth mode, selecting the burst length increment, and enabling bandwidth underflow and overflow detection.

SWU\_CTLn: Control Register n - R/W

Reset = 0x0000 0000

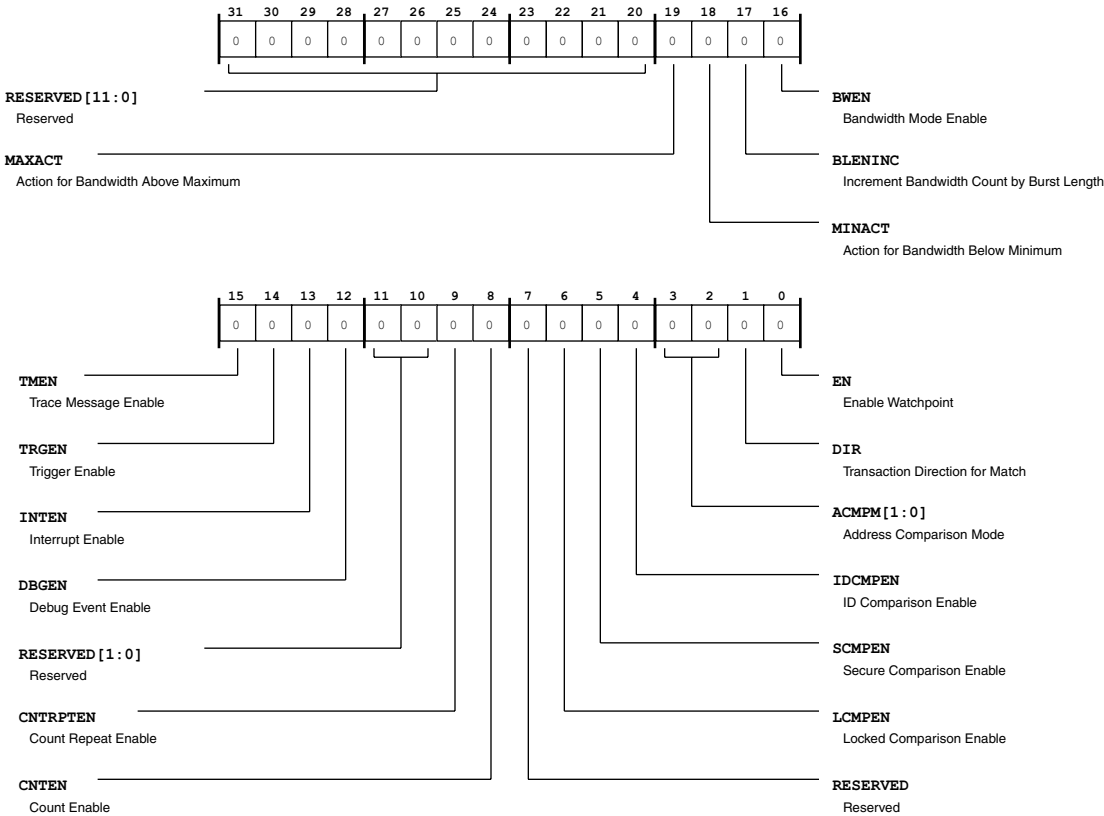


Figure 36-5: SWU\_CTLn Register Diagram

Table 36-8: SWU\_CTLn Register Fields

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                 |
|---------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 31:20   | RESERVED | Reserved.                                                                                                                                                                               |
| 19      | MAXACT   | Action for Bandwidth Above Maximum.<br>Each SWU_CTLn.MAXACT bit determines whether a watchpoint group takes action on bandwidth overflow. This feature is only valid in bandwidth mode. |
|         |          | 0      No Action                                                                                                                                                                        |
|         |          | 1      Take Action                                                                                                                                                                      |

**Table 36-8:** SWU\_CTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 18      | MINACT   | Action for Bandwidth Below Minimum.<br>Each SWU_CTLn.MINACT bit determines whether a watchpoint group takes action on bandwidth underflow. This feature is only valid in bandwidth mode.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0      No Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|         |          | 1      Take Action                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 17      | BLENINC  | Increment Bandwidth Count by Burst Length.<br>Each SWU_CTLn.BLENINC bit controls how a watchpoint group's bandwidth count is incremented in the SWU_CURn register's SWU_CURn.CURBW field. If the SWU_CTLn.BLENINC bit is cleared (= 0), the SWU increments the bandwidth count by 1 for each matching transaction. If the SWU_CTLn.BLENINC bit is set (=1), the SWU increments the bandwidth count by the burst length of the transaction for each matching transaction. This feature is only valid for bandwidth mode (SWU_CTLn.BWEN bit == 1).<br>Note that if the address range match is enabled (SWU_CTLn.ACMPM bits) and if any address of a burst falls within the address range, the SWU_CURn.CURBW field is incremented by the burst length even if some of the burst address fall outside of the range.<br>Also, note that the burst size of the transaction is not included in the increment, only the burst length of the transaction. This increment operation provides an approximate (not exact) number of bus cycles consumed during the bandwidth. |
|         |          | 0      Increment by 1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|         |          | 1      Burst Length Increment for Bandwidth Count                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Table 36-8:** SWU\_CTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 16      | BWEN     | Bandwidth Mode Enable.<br>Each SWU_CTLn.BWEN bit controls whether a watchpoint group operates in watchpoint mode or bandwidth mode. In watchpoint mode, the SWU_CTLn.CNTEN and (optionally) SWU_CTLn.CNTRPTEN registers control usage of the cycle count for watchpoint group operations. In bandwidth mode, the SWU_CTLn.BLENINC, SWU_TARGn, and SWU_HISTn registers control usage of watchpoint matches for watchpoint group operations. |
|         |          | 0 Watchpoint Mode                                                                                                                                                                                                                                                                                                                                                                                                                          |
|         |          | 1 Bandwidth Mode                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 15      | TMEN     | Trace Message Enable.<br>Each SWU_CTLn.TMEN bit controls whether a match for a watchpoint group generates a trace message event. This feature is valid in both bandwidth and watchpoint modes.                                                                                                                                                                                                                                             |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 14      | TRGEN    | Trigger Enable.<br>Each SWU_CTLn.TRGEN bit controls whether a match for a watchpoint group generates a trigger event. This feature is valid in both bandwidth and watchpoint modes.                                                                                                                                                                                                                                                        |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 13      | INTEN    | Interrupt Enable.<br>Each SWU_CTLn.INTEN bit controls whether a match for a watchpoint group generates an interrupt. This feature is valid in both bandwidth and watchpoint modes.                                                                                                                                                                                                                                                         |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                                                   |

**Table 36-8:** SWU\_CTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 12      | DBGEN    | Debug Event Enable.<br>Each SWU_CTLn.DBGEN bit controls debug event comparison for a watchpoint group, permitting matches based on debug status.                                                                                                                                                                                                                                                                                                                                                                                                           |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 11:10   | RESERVED | Reserved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 9       | CNTRPTEN | Count Repeat Enable.<br>Each SWU_CTLn.CNTRPTEN bit controls whether the watchpoint group's cycle count is reloaded and repeated after cycle countdown. If the SWU_CTLn register's SWU_CTLn.CNTRPTEN bit is set, the SWU_CURn register's SWU_CURn.CURCNT field is reloaded from SWU_CNTn register's SWU_CNTn.COUNT field, and the countdown starts again. If SWU_CTLn.CNTRPTEN bit is cleared, the expired count remains zero, and no further events are signalled. (See the SWU_CTLn.CNTEN bit description for information regarding the countdown setup.) |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 8       | CNTEN    | Count Enable.<br>Each SWU_CTLn.CNTEN bit controls whether the cycle count in the watchpoint group's SWU_CNTn register is decremented each cycle until it reaches zero. This feature is only valid in watchpoint mode (SWU_CTLn.BWEN bit == 0). When the count reaches zero, any enabled watchpoint events are triggered. (See the SWU_CTLn.CNTRPTEN bit description for optional actions at that may occur at the end of the countdown.)                                                                                                                   |
|         |          | 0                      Disable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|         |          | 1                      Enable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 7       | RESERVED | Reserved.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**Table 36-8:** SWU\_CTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                    |
|---------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 6       | LCMPEN   | Locked Comparison Enable.<br>Each SWU_CTLn.LCMPEN bit controls locked comparison operation of an SWU watchpoint group, permitting matches based on lock status.                                                                                                                                                                            |
|         |          | 0 Match on all transaction                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Match only locked transactions                                                                                                                                                                                                                                                                                                           |
| 5       | SCMPEN   | Secure Comparison Enable.<br>Each SWU_CTLn.SCMPEN bit controls secure transaction comparison operation of an SWU watchpoint group, permitting matches based on transaction security.                                                                                                                                                       |
|         |          | 0 Match on all transaction                                                                                                                                                                                                                                                                                                                 |
|         |          | 1 Match only secure transactions                                                                                                                                                                                                                                                                                                           |
| 4       | IDCMPEN  | ID Comparison Enable.<br>Each SWU_CTLn.IDCMPEN bit controls the ID comparison operation of an SWU watchpoint group. The ID match is based on comparison with the value in the SWU_IDn register.                                                                                                                                            |
| 3:2     | ACMPM    | Address Comparison Mode.<br>Each set of SWU_CTLn.ACMPM bits control the address comparison operation of an SWU watchpoint group. The address within range for comparison is defined as (SWU_LAn register ≤ address < SWU_UAn register). The address outside range for comparison is defined as (address < SWU_LAn) or (SWU_UAn ≤ address). |
|         |          | 0 No address comparison                                                                                                                                                                                                                                                                                                                    |
|         |          | 1 Exact match on LAn                                                                                                                                                                                                                                                                                                                       |
|         |          | 2 Match on address within range                                                                                                                                                                                                                                                                                                            |
|         |          | 3 Match on address outside range                                                                                                                                                                                                                                                                                                           |

**Table 36-8:** SWU\_CTLn Register Fields (Continued)

| Bit No. | Bit Name | Description/Enumeration                                                                                                                                                                                                                                                                                                                                                                                          |
|---------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1       | DIR      | Transaction Direction for Match.<br>Each SWU_CTLn.DIR bit determines whether the SWU check reads or writes for watchpoint matches.                                                                                                                                                                                                                                                                               |
|         |          | 0 Match on reads only                                                                                                                                                                                                                                                                                                                                                                                            |
|         |          | 1 Match on writes only                                                                                                                                                                                                                                                                                                                                                                                           |
| 0       | EN       | Enable Watchpoint.<br>Each SWU_CTLn.EN bit controls the operation of one SWU watchpoint group. Clearing the SWU_CTLn.EN bit halts the execution of watchpoint or bandwidth tracking operations in the watchpoint group without resetting status or configuration registers. Setting the SWU_CTLn.EN bit enables the SWU watchpoint group to begin or resume operation with the current configuration and status. |
|         |          | 0 Disable                                                                                                                                                                                                                                                                                                                                                                                                        |
|         |          | 1 Enable                                                                                                                                                                                                                                                                                                                                                                                                         |

## Lower Address Register n

The SWU lower address registers (SWU\_LAn) contain each watchpoint group's lower address for address match comparison. In exact match on SWU\_LAn address mode (SWU\_CTLn.ACMPM bits = 01), the watchpoint group uses only this address for match comparison.

**SWU\_LAn: Lower Address Register n - R/W**

Reset = 0x0000 0000

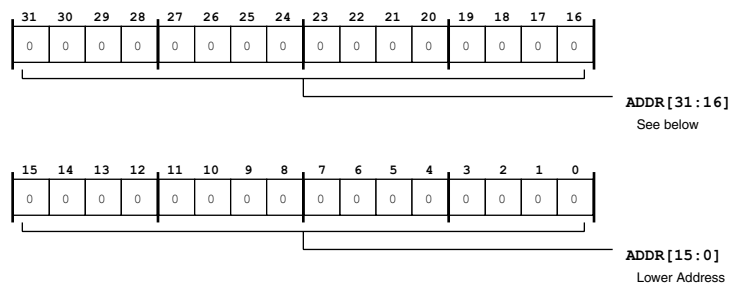


**Figure 36-6:** SWU\_LAn Register Diagram

Table 36-9: SWU\_LAn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | ADDR     | Lower Address.          |

Upper Address Register n

The SWU upper address registers (SWU\_UAn) contain each watchpoint group's upper address for address match comparison. In exact match on SWU\_LAn address mode (SWU\_CTLn.ACMPM bits =01), the SWU\_UAn is not used for match comparison.

SWU\_UAn: Upper Address Register n - R/W

Reset = 0x0000 0000

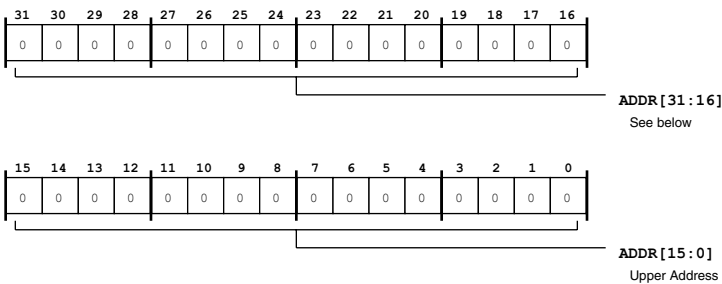


Figure 36-7: SWU\_UAn Register Diagram

Table 36-10: SWU\_UAn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:0    | ADDR     | Upper Address.          |

ID Register n

The SWU ID registers (SWU\_IDn) contain a 16-bit ID field (SWU\_IDn.ID) and a 16-bit ID mask field (SWU\_IDn.IDMASK) that watchpoint groups use for ID comparison. The ID on the bus is AND'ed with the SWU\_IDn.IDMASK field, then the watchpoint group compares the result against the SWU\_IDn.ID field.



**SWU\_IDn: ID Register n - R/W**

Reset = 0x0000 0000

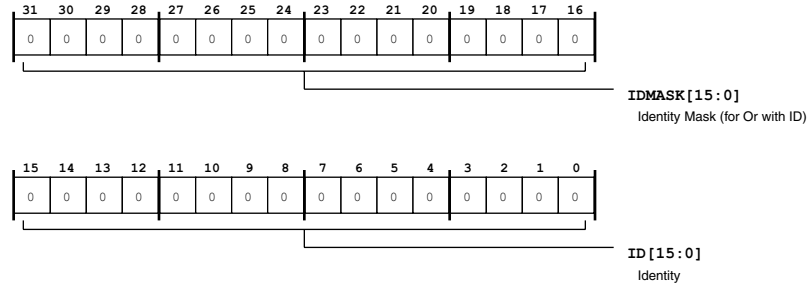


**Figure 36-8:** SWU\_IDn Register Diagram

**Table 36-11:** SWU\_IDn Register Fields

| Bit No. | Bit Name | Description/Enumeration         |
|---------|----------|---------------------------------|
| 31:16   | IDMASK   | Identity Mask (for Or with ID). |
| 15:0    | ID       | Identity.                       |

## Count Register n

The SWU count registers (SWU\_CNTn) contain a 16-bit count field (SWU\_CNTn.COUNT) whose usage differs depending on the mode of the watchpoint group. In bandwidth mode, the SWU\_CNTn.COUNT field value defines the number of clock cycles in a bandwidth period. In watchpoint mode, when the cycle count is enabled, the SWU\_CNTn.COUNT field value determines how many matches occur before the watchpoint group takes action.

**SWU\_CNTn: Count Register n - R/W**

Reset = 0x0000 0000

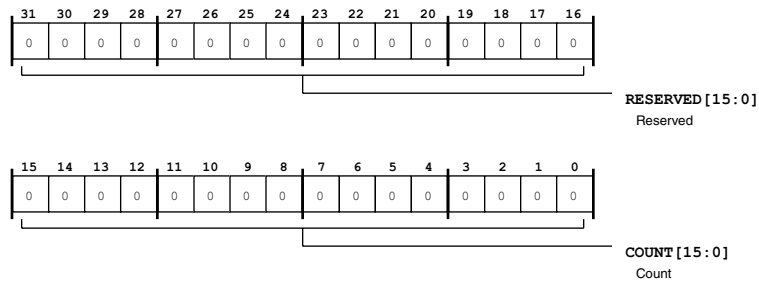


**Figure 36-9:** SWU\_CNTn Register Diagram

**Table 36-12:** SWU\_CNTn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | RESERVED | Reserved.               |
| 15:0    | COUNT    | Count.                  |

Target Register n

The SWU target registers (SWU\_TARGn) contain a minimum value field (SWU\_TARGn.BWMIN) and maximum value field (SWU\_TARGn.BWMAX) of bandwidth targets used by watchpoint groups in bandwidth mode. When the bandwidth period expires, if the current bandwidth value (SWU\_CURn register, SWU\_CURn.CURBW bits) is below the minimum target or above the maximum target, the watchpoint group takes action as enabled by the SWU\_CTLn register's SWU\_CTLn.MINACT or SWU\_CTLn.MAXACT bits.

In bandwidth mode, note that the watchpoint group increments its count of either data bus transactions or address bus transactions (bursts) as selected by the SWU\_CTLn.BLENINC bit. Keep this mode selection in mind when programming the bandwidth target values.

SWU\_TARGn: Target Register n - R/W

Reset = 0x0000 0000

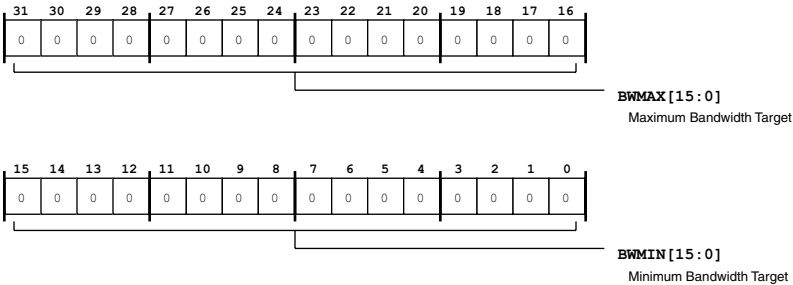


**Figure 36-10:** SWU\_TARGn Register Diagram

**Table 36-13:** SWU\_TARGn Register Fields

| Bit No. | Bit Name | Description/Enumeration   |
|---------|----------|---------------------------|
| 31:16   | BWMAX    | Maximum Bandwidth Target. |
| 15:0    | BWMIN    | Minimum Bandwidth Target. |

## Bandwidth History Register n

The SWU bandwidth history registers (SWU\_HISTn) contain data copied from a watchpoint group's current bandwidth value (SWU\_CURn register, SWU\_CURn.CURBW bits) at the end of the last two watchpoint periods. At the end of each watchpoint period, the SWU copies the previous bandwidth value from the SWU\_HISTn.BWHIST0 field to the SWU\_HISTn.BWHIST1 field and copies the new bandwidth value from the SWU\_CURn.CURBW field to the SWU\_HISTn.BWHIST0 field.

**SWU\_HISTn: Bandwidth History Register n - R/W**

Reset = 0x0000 0000

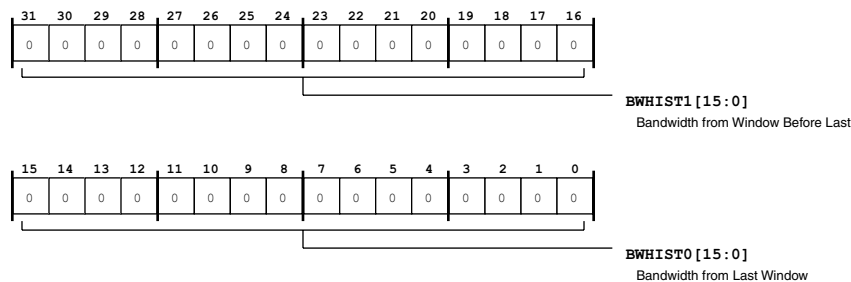


**Figure 36-11:** SWU\_HISTn Register Diagram

**Table 36-14:** SWU\_HISTn Register Fields

| Bit No. | Bit Name | Description/Enumeration            |
|---------|----------|------------------------------------|
| 31:16   | BWHIST1  | Bandwidth from Window Before Last. |
| 15:0    | BWHIST0  | Bandwidth from Last Window.        |

## Current Register n

The SWU current register (SWU\_CURn) operation varies depending whether the watchpoint group is in bandwidth mode or watchpoint mode. In both modes, the watchpoint count begins when the SWU loads the register's SWU\_CURn.CURCNT field from the SWU\_CNTn register's SWU\_CNTn.COUNT field when the watchpoint count is enabled (SWU\_CTLn register, SWU\_CTLn.CNTEN bit =1).

In bandwidth mode, the current count field (SWU\_CURn.CURCNT) contains the cycle count remaining within the current watchpoint period. The SWU decrements this value every cycle until the count reaches zero. At that point, the SWU reloads the SWU\_CURn.CURCNT field from SWU\_CNTn register's SWU\_CNTn.COUNT field. In bandwidth mode, the current bandwidth field (SWU\_CURn.CURBW) contains the count of watchpoint matches (bandwidth) accumulated in the current watchpoint period.

In watchpoint mode, the current count field (SWU\_CURn.CURCNT) contains the watchpoint match count remaining within the current watchpoint period. The SWU decrements this value with every watchpoint match until the count reaches zero. At that point, the SWU reloads the SWU\_CURn.CURCNT field from

SWU\_CNTn register's SWU\_CNTn.COUNT field if the SWU\_CTLn register's SWU\_CTLn.CNTRPTEN bit is set (=1).  
In watchpoint mode, the current bandwidth field (SWU\_CURn.CURBW) is undefined.

SWU\_CURn: Current Register n - R/W

Reset = 0x0000 0000

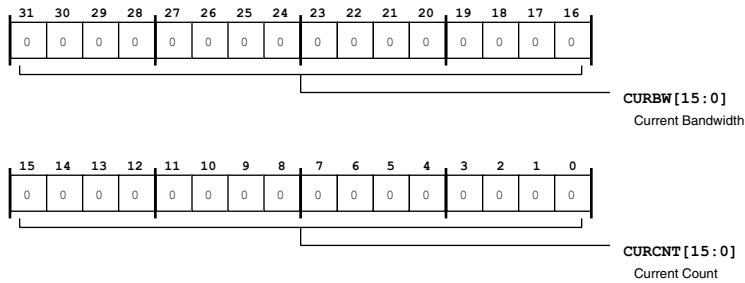


Figure 36-12: SWU\_CURn Register Diagram

Table 36-15: SWU\_CURn Register Fields

| Bit No. | Bit Name | Description/Enumeration |
|---------|----------|-------------------------|
| 31:16   | CURBW    | Current Bandwidth.      |
| 15:0    | CURCNT   | Current Count.          |

# I Index

## A

- Abort Acknowledge 1 Register, CAN (CAN\_AA1), 21-35
- Abort Acknowledge 2 Register, CAN (CAN\_AA2), 21-42
- Acceptance Mask (H) Register, CAN (CAN\_AMnnH), 21-76
- Acceptance Mask (L) Register, CAN (CAN\_AMnnL), 21-76
- Access Control Core 0 Register, L2CTL (L2CTL\_ACTL\_C0), 10-12
- Access Control Core 1 Register, L2CTL (L2CTL\_ACTL\_C1), 10-15
- Access Control System Register, L2CTL (L2CTL\_ACTL\_SYS), 10-17
- ACM Completed Event Interrupt Mask Register, ACM (ACM\_EVMSK), 27-34
- ACM Control Register, ACM (ACM\_CTL), 27-25
- ACM Event Status Register, ACM (ACM\_EVSTAT), 27-32
- ACM Eventn Control Register, ACM (ACM\_EVCTLn), 27-40
- ACM Eventn Order Register, ACM (ACM\_EVORDn), 27-41
- ACM Eventn Time Register, ACM (ACM\_EVTIMEn), 27-41
- ACM Missed Event Interrupt Mask Register, ACM (ACM\_MEVMSK), 27-38
- ACM Missed Event Status Register, ACM (ACM\_MEVSTAT), 27-36
- ACM Status Register, ACM (ACM\_STAT), 27-30
- ACM Timer 0 Register, ACM (ACM\_TMR0), 27-42
- ACM Timer 1 Register, ACM (ACM\_TMR1), 27-43
- ACM Timing Configuration 0 Register, ACM (ACM\_TC0), 27-28
- ACM Timing Configuration 1 Register, ACM (ACM\_TC1), 27-29
- ACM0 Event Complete (Masters), 8-3
- ACM0 Event Complete interrupt, 7-5, 27-19
- ACM0 Event Miss interrupt, 7-5, 27-19
- ACM0 Trigger Input 2 (Slaves), 8-8
- ACM0 Trigger Input 3 (Slaves), 8-8
- ACM\_CTL (ACM Control Register, ACM), 27-25
- ACM\_EVCTLn (ACM Eventn Control Register, ACM), 27-40
- ACM\_EVMSK (ACM Completed Event Interrupt Mask Register, ACM), 27-34
- ACM\_EVORDn (ACM Eventn Order Register, ACM), 27-41
- ACM\_EVSTAT (ACM Event Status Register, ACM), 27-32
- ACM\_EVTIMEn (ACM Eventn Time Register, ACM), 27-41
- ACM\_MEVMSK (ACM Missed Event Interrupt Mask Register, ACM), 27-38
- ACM\_MEVSTAT (ACM Missed Event Status Register, ACM), 27-36
- ACM\_STAT (ACM Status Register, ACM), 27-30
- ACM\_TC0 (ACM Timing Configuration 0 Register, ACM), 27-28
- ACM\_TC1 (ACM Timing Configuration 1 Register, ACM), 27-29
- ACM\_TMR0 (ACM Timer 0 Register, ACM), 27-42
- ACM\_TMR1 (ACM Timer 1 Register, ACM), 27-43
- ACU Configuration, PVP (PVP\_ACU\_CFG), 30-152
- ACU Control, PVP (PVP\_ACU\_CTL), 30-154
- ACU Lower Sat Threshold Min, PVP (PVP\_ACU\_MIN), 30-159
- ACU PROD Constant, PVP (PVP\_ACU\_FACTOR), 30-158
- ACU Shift Constant, PVP (PVP\_ACU\_SHIFT), 30-158
- ACU SUM Constant, PVP (PVP\_ACU\_OFFSET), 30-157
- ACU Upper Sat Threshold Max, PVP (PVP\_ACU\_MAX), 30-159

Arbitration Read Channel Master Interface n Register, SCB (SCB\_ARBRn), 2-22  
 Arbitration Write Channel Master Interface n Register, SCB (SCB\_ARBWn), 2-23  
 Argument Register, RSI (RSI\_ARG), 24-46

## B

Bandwidth History Register n, SWU (SWU\_HISTn), 36-25  
 Bandwidth Limit Count Current, DMA (DMA\_BWLCNT\_CUR), 13-69  
 Bandwidth Limit Count, DMA (DMA\_BWLCNT), 13-68  
 Bandwidth Monitor Count Current, DMA (DMA\_BWMCNT\_CUR), 13-70  
 Bandwidth Monitor Count, DMA (DMA\_BWMCNT), 13-69  
 Bank 0 Control Register, SMC (SMC\_B0CTL), 9-26  
 Bank 0 Extended Timing Register, SMC (SMC\_B0ETIM), 9-32  
 Bank 0 Timing Register, SMC (SMC\_B0TIM), 9-30  
 Bank 1 Control Register, SMC (SMC\_B1CTL), 9-34  
 Bank 1 Extended Timing Register, SMC (SMC\_B1ETIM), 9-40  
 Bank 1 Timing Register, SMC (SMC\_B1TIM), 9-38  
 Bank 2 Control Register, SMC (SMC\_B2CTL), 9-42  
 Bank 2 Extended Timing Register, SMC (SMC\_B2ETIM), 9-48  
 Bank 2 Timing Register, SMC (SMC\_B2TIM), 9-46  
 Bank 3 Control Register, SMC (SMC\_B3CTL), 9-50  
 Bank 3 Extended Timing Register, SMC (SMC\_B3ETIM), 9-56  
 Bank 3 Timing Register, SMC (SMC\_B3TIM), 9-54  
 Battery Charging Control Register, USB (USB\_BAT\_CHG), 22-193  
 Block Size Register, RSI (RSI\_BLKSZ), 24-63  
 Boot Acknowledge Timeout Register, RSI (RSI\_BACK\_TOUT), 24-62  
 Boot Code Register, RCU (RCU\_BCODE), 33-14, 34-72  
 Boot Timing Counter Register, RSI (RSI\_BOOT\_TCNT), 24-62  
 Broadcast Delay Register, TIMER (TIMER\_BCAST\_DLY), 15-42  
 Broadcast Period Register, TIMER

(TIMER\_BCAST\_PER), 15-40  
 Broadcast Width Register, TIMER (TIMER\_BCAST\_WID), 15-41  
 BV Conversion Component Register, PIXC (PIXC\_CONBV), 32-48

## C

CAN Master Control Register, CAN (CAN\_CTL), 21-63  
 CAN0 Receive interrupt, 7-5, 21-4  
 CAN0 Status interrupt, 7-5, 21-4  
 CAN0 Transmit interrupt, 7-5, 21-4  
 CAN\_AA1 (Abort Acknowledge 1 Register, CAN), 21-35  
 CAN\_AA2 (Abort Acknowledge 2 Register, CAN), 21-42  
 CAN\_AMnnH (Acceptance Mask (H) Register, CAN), 21-76  
 CAN\_AMnnL (Acceptance Mask (L) Register, CAN), 21-76  
 CAN\_CEC (Error Counter Register, CAN), 21-54  
 CAN\_CLK (Clock Register, CAN), 21-46  
 CAN\_CTL (CAN Master Control Register, CAN), 21-63  
 CAN\_DBG (Debug Register, CAN), 21-49  
 CAN\_ESR (Error Status Register, CAN), 21-69  
 CAN\_EWR (Error Counter Warning Level Register, CAN), 21-68  
 CAN\_GIF (Global CAN Interrupt Flag Register, CAN), 21-61  
 CAN\_GIM (Global CAN Interrupt Mask Register, CAN), 21-58  
 CAN\_GIS (Global CAN Interrupt Status Register, CAN), 21-55  
 CAN\_INT (Interrupt Pending Register, CAN), 21-66  
 CAN\_MBIM1 (Mailbox Interrupt Mask 1 Register, CAN), 21-37  
 CAN\_MBIM2 (Mailbox Interrupt Mask 2 Register, CAN), 21-45  
 CAN\_MBnn\_DATA0 (Mailbox Word 0 Register, CAN), 21-77  
 CAN\_MBnn\_DATA1 (Mailbox Word 1 Register, CAN), 21-78  
 CAN\_MBnn\_DATA2 (Mailbox Word 2 Register, CAN), 21-79

- CAN\_MBnn\_DATA3 (Mailbox Word 3 Register, CAN), 21-79
- CAN\_MBnn\_ID0 (Mailbox ID 0 Register, CAN), 21-81
- CAN\_MBnn\_ID1 (Mailbox ID 1 Register, CAN), 21-82
- CAN\_MBnn\_LENGTH (Mailbox Length Register, CAN), 21-80
- CAN\_MBnn\_TIMESTAMP (Mailbox Timestamp Register, CAN), 21-80
- CAN\_MBRIF1 (Mailbox Receive Interrupt Flag 1 Register, CAN), 21-37
- CAN\_MBRIF2 (Mailbox Receive Interrupt Flag 2 Register, CAN), 21-44
- CAN\_MBTD (Temporary Mailbox Disable Register, CAN), 21-67
- CAN\_MBTIF1 (Mailbox Transmit Interrupt Flag 1 Register, CAN), 21-36
- CAN\_MBTIF2 (Mailbox Transmit Interrupt Flag 2 Register, CAN), 21-43
- CAN\_MC1 (Mailbox Configuration 1 Register, CAN), 21-32
- CAN\_MC2 (Mailbox Configuration 2 Register, CAN), 21-39
- CAN\_MD1 (Mailbox Direction 1 Register, CAN), 21-32
- CAN\_MD2 (Mailbox Direction 2 Register, CAN), 21-40
- CAN\_OPSS1 (Overwrite Protection/Single Shot Transmission 1 Register, CAN), 21-39
- CAN\_OPSS2 (Overwrite Protection/Single Shot Transmission 2 Register, CAN), 21-46
- CAN\_RFH1 (Remote Frame Handling 1 Register, CAN), 21-38
- CAN\_RFH2 (Remote Frame Handling 2 Register, CAN), 21-45
- CAN\_RML1 (Receive Message Lost 1 Register, CAN), 21-36
- CAN\_RML2 (Receive Message Lost 2 Register, CAN), 21-43
- CAN\_RMP1 (Receive Message Pending 1 Register, CAN), 21-35
- CAN\_RMP2 (Receive Message Pending 2 Register, CAN), 21-42
- CAN\_STAT (Status Register, CAN), 21-51
- CAN\_TA1 (Transmission Acknowledge 1 Register, CAN), 21-34
- CAN\_TA2 (Transmission Acknowledge 2 Register, CAN), 21-41
- CAN\_TIMING (Timing Register, CAN), 21-47
- CAN\_TRR1 (Transmission Request Reset 1 Register, CAN), 21-34
- CAN\_TRR2 (Transmission Request Reset 2 Register, CAN), 21-41
- CAN\_TRS1 (Transmission Request Set 1 Register, CAN), 21-33
- CAN\_TRS2 (Transmission Request Set 2 Register, CAN), 21-40
- CAN\_UCCNF (Universal Counter Configuration Mode Register, CAN), 21-73
- CAN\_UCCNT (Universal Counter Register, CAN), 21-72
- CAN\_UCRC (Universal Counter Reload/Capture Register, CAN), 21-72
- CGU0 Error interrupt, 3-2, 7-9
- CGU0 Event (Masters), 3-2, 8-2
- CGU0 Event interrupt, 3-2, 7-3
- CGU\_CLKOUTSEL (CLKOUT Select Register, CGU), 3-20
- CGU\_CTL (Control Register, CGU), 3-11
- CGU\_DIV (Divisor Register, CGU), 3-17
- CGU\_STAT (Status Register, CGU), 3-13
- Channel A Control Register, PWM (PWM\_ACTL), 18-76
- Channel A Delay Register, PWM (PWM\_DLYA), 18-73
- Channel A-High Duty-0 Register, PWM (PWM\_AH0), 18-80
- Channel A-High Duty-1 Register, PWM (PWM\_AH1), 18-80
- Channel A-Low Duty-0 Register, PWM (PWM\_AL0), 18-82
- Channel A-Low Duty-1 Register, PWM (PWM\_AL1), 18-82
- Channel B Control Register, PWM (PWM\_BCTL), 18-83
- Channel B Delay Register, PWM (PWM\_DLYB), 18-74
- Channel B-High Duty-0 Register, PWM (PWM\_BH0), 18-87

- Channel B-High Duty-1 Register, PWM (PWM\_BH1), 18-87
- Channel B-Low Duty-0 Register, PWM (PWM\_BL0), 18-88
- Channel B-Low Duty-1 Register, PWM (PWM\_BL1), 18-89
- Channel C Control Register, PWM (PWM\_CCTL), 18-90
- Channel C Delay Register, PWM (PWM\_DLYC), 18-75
- Channel C-High Pulse Duty Register 0, PWM (PWM\_CH0), 18-93
- Channel C-High Pulse Duty Register 1, PWM (PWM\_CH1), 18-93
- Channel C-Low Duty-1 Register, PWM (PWM\_CL1), 18-95
- Channel C-Low Pulse Duty Register 0, PWM (PWM\_CL0), 18-95
- Channel Config Register, PWM (PWM\_CHANCFG), 18-44
- Channel D Control Register, PWM (PWM\_DCTL), 18-96
- Channel D Delay Register, PWM (PWM\_DLYD), 18-76
- Channel D-High Duty-0 Register, PWM (PWM\_DH0), 18-99
- Channel D-High Pulse Duty Register 1, PWM (PWM\_DH1), 18-99
- Channel D-Low Pulse Duty Register 0, PWM (PWM\_DL0), 18-101
- Channel D-Low Pulse Duty Register 1, PWM (PWM\_DL1), 18-101
- Chirp Timeout Register, USB (USB\_CT\_UCH), 22-181
- Chop Configuration Register, PWM (PWM\_CHOPCFG), 18-66
- Clipping Register for EVEN (Luma) Data, EPPI (EPPI\_EVENCLIP), 31-94
- Clipping Register for ODD (Chroma) Data, EPPI (EPPI\_ODDCLIP), 31-93
- CLKOUT Select Register, CGU (CGU\_CLKOUTSEL), 3-20
- Clock Divide Register, EPPI (EPPI\_CLKDIV), 31-76
- Clock Divider Value, LP (LP\_DIV), 28-23
- Clock Rate Register, SPI (SPI\_CLK), 25-37
- Clock Rate Register, UART (UART\_CLK), 19-40
- Clock Register, CAN (CAN\_CLK), 21-46
- CNT0 Status (Masters), 8-3, 17-3
- CNT0 Status interrupt, 7-4, 17-3
- CNT\_CFG (Configuration Register, CNT), 17-15
- CNT\_CMD (Command Register, CNT), 17-22
- CNT\_CNTR (Counter Register, CNT), 17-26
- CNT\_DEBNCE (Debounce Register, CNT), 17-23
- CNT\_IMSK (Interrupt Mask Register, CNT), 17-18
- CNT\_MAX (Maximum Count Register, CNT), 17-27
- CNT\_MIN (Minimum Count Register, CNT), 17-27
- CNT\_STAT (Status Register, CNT), 17-20
- CNVn Coefficient 0,4, PVP (PVP\_CNVn\_C04), 30-179
- CNVn Coefficient 1,4, PVP (PVP\_CNVn\_C14), 30-180
- CNVn Coefficient 2,4, PVP (PVP\_CNVn\_C24), 30-182
- CNVn Coefficient 3,4, PVP (PVP\_CNVn\_C34), 30-184
- CNVn Coefficient 4,4, PVP (PVP\_CNVn\_C44), 30-186
- CNVn Coefficients 0,0 and 0,1, PVP (PVP\_CNVn\_C00C01), 30-177
- CNVn Coefficients 0,2 and 0,3, PVP (PVP\_CNVn\_C02C03), 30-178
- CNVn Coefficients 1,0 and 1,1, PVP (PVP\_CNVn\_C10C11), 30-179
- CNVn Coefficients 1,2 and 1,3, PVP (PVP\_CNVn\_C12C13), 30-180
- CNVn Coefficients 2,0 and 2,1, PVP (PVP\_CNVn\_C20C21), 30-181
- CNVn Coefficients 2,2 and 2,3, PVP (PVP\_CNVn\_C22C23), 30-182
- CNVn Coefficients 3,0 and 3,1, PVP (PVP\_CNVn\_C30C31), 30-183
- CNVn Coefficients 3,2 and 3,3, PVP (PVP\_CNVn\_C32C33), 30-183
- CNVn Coefficients 4,0 and 4,1, PVP (PVP\_CNVn\_C40C41), 30-185
- CNVn Coefficients 4,2 and 4,3, PVP (PVP\_CNVn\_C42C43), 30-185
- CNVn Configuration, PVP (PVP\_CNVn\_CFG), 30-175
- CNVn Control, PVP (PVP\_CNVn\_CTL), 30-176



- CNVn Scaling Factor, PVP (PVP\_CNVn\_SCALE), 30-187
- Command Register, CNT (CNT\_CMD), 17-22
- Command Register, RSI (RSI\_CMD), 24-46
- Common Interrupts Enable Register, USB (USB\_IEN), 22-107
- Common Interrupts Register, USB (USB\_IRQ), 22-105
- Configuration Register, CNT (CNT\_CFG), 17-15
- Configuration Register, DMA (DMA\_CFG), 13-48
- Configuration Register, DMC (DMC\_CFG), 11-28
- Configuration Register, RSI (RSI\_CFG), 24-70
- Control Register 2, EPPI (EPPI\_CTL2), 31-96
- Control Register n, SWU (SWU\_CTLn), 36-16
- Control Register, CGU (CGU\_CTL), 3-11
- Control Register, CRC (CRC\_CTL), 12-29
- Control Register, DMC (DMC\_CTL), 11-16
- Control Register, DPM (DPM\_CTL), 5-13
- Control Register, EPPI (EPPI\_CTL), 31-77
- Control Register, L2CTL (L2CTL\_CTL), 10-8
- Control Register, LP (LP\_CTL), 28-19
- Control Register, PIXC (PIXC\_CTL), 32-37
- Control Register, PWM (PWM\_CTL), 18-40
- Control Register, RCU (RCU\_CTL), 33-6
- Control Register, RSI (RSI\_CTL), 24-44
- Control Register, SDU (SDU\_CTL), 35-14
- Control Register, SPI (SPI\_CTL), 25-25
- Control Register, SPU (SPU\_CTL), 4-8
- Control Register, TWI (TWI\_CTL), 20-19
- Control Register, UART (UART\_CTL), 19-26
- Control Register, WDOG (WDOG\_CTL), 16-4
- Control, PVP (PVP\_CTL), 30-111
- Conversion Bias Register, PIXC (PIXC\_CCBIAS), 32-49
- Core 0 Double Fault interrupt, 7-3
- Core 0 Hardware Error interrupt, 7-3
- Core 0 Unhandled NMI or L1 Memory Parity Error interrupt, 7-3
- Core 0 Wakeup Input 0 (Slaves), 8-10
- Core 0 Wakeup Input 1 (Slaves), 8-10
- Core 0 Wakeup Input 2 (Slaves), 8-10
- Core 0 Wakeup Input 3 (Slaves), 8-10
- Core 1 Double Fault interrupt, 7-3
- Core 1 Hardware Error interrupt, 7-3
- Core 1 Unhandled NMI or L1 Memory Parity Error interrupt, 7-3
- Core 1 Wakeup Input 0 (Slaves), 8-10
- Core 1 Wakeup Input 1 (Slaves), 8-10
- Core 1 Wakeup Input 2 (Slaves), 8-10
- Core 1 Wakeup Input 3 (Slaves), 8-10
- Core Clock Buffer Disable Register, DPM (DPM\_CCBF\_DIS), 5-18
- Core Clock Buffer Enable Register, DPM (DPM\_CCBF\_EN), 5-19
- Core Clock Buffer Status Register, DPM (DPM\_CCBF\_STAT), 5-20
- Core Clock Buffer Status Sticky Register, DPM (DPM\_CCBF\_STAT\_STKY), 5-20
- Core Pending Register n, SEC (SEC\_CPNDn), 7-23
- Core Reset Control Register, RCU (RCU\_CRCTL), 33-10
- Core Reset Status Register, RCU (RCU\_CRSTAT), 33-11
- Count Register n, SWU (SWU\_CNTn), 36-23
- Count Register, WDOG (WDOG\_CNT), 16-5
- Counter Register, CNT (CNT\_CNTR), 17-26
- CRC Current Result Register, CRC (CRC\_RESULT\_CUR), 12-44
- CRC Final Result Register, CRC (CRC\_RESULT\_FIN), 12-43
- CRC0 Datacount expiration interrupt, 7-7, 12-4
- CRC0 Error interrupt, 7-7, 12-4
- CRC1 Datacount expiration interrupt, 7-7, 12-4
- CRC1 Error interrupt, 7-7, 12-4
- CRC\_COMP (Data Compare Register, CRC), 12-34
- CRC\_CTL (Control Register, CRC), 12-29
- CRC\_DCNT (Data Word Count Register, CRC), 12-33
- CRC\_DCNTCAP (Data Count Capture Register, CRC), 12-42
- CRC\_DCNTRLD (Data Word Count Reload Register, CRC), 12-34
- CRC\_DFIFO (Data FIFO Register, CRC), 12-36
- CRC\_FILLVAL (Fill Value Register, CRC), 12-35
- CRC\_INEN (Interrupt Enable Register, CRC), 12-36
- CRC\_INEN\_CLR (Interrupt Enable Clear Register, CRC), 12-38
- CRC\_INEN\_SET (Interrupt Enable Set Register, CRC), 12-37
- CRC\_POLY (Polynomial Register, CRC), 12-39
- CRC\_RESULT\_CUR (CRC Current Result Register,

CRC), 12-44  
 CRC\_RESULT\_FIN (CRC Final Result Register, CRC), 12-43  
 CRC\_REVID (Revision ID Register, CRC), 12-44  
 CRC\_STAT (Status Register, CRC), 12-40  
 Current Address, DMA (DMA\_ADDR\_CUR), 13-61  
 Current Count(1D) or intra-row XCNT (2D), DMA (DMA\_XCNT\_CUR), 13-67  
 Current Descriptor Pointer, DMA (DMA\_DSCPTR\_CUR), 13-60  
 Current Register n, SWU (SWU\_CURn), 36-26  
 Current Row Count (2D only), DMA (DMA\_YCNT\_CUR), 13-67

## D

Data Compare Register, CRC (CRC\_COMP), 12-34  
 Data Control Register, RSI (RSI\_DATA\_CTL), 24-52  
 Data Count Capture Register, CRC (CRC\_DCNTCAP), 12-42  
 Data Count Register, RSI (RSI\_DATA\_CNT), 24-53  
 Data FIFO Register, CRC (CRC\_DFIFO), 12-36  
 Data FIFO Register, RSI (RSI\_FIFO), 24-64  
 Data Interrupt Latch Register, TIMER (TIMER\_DATA\_ILAT), 15-36  
 Data Interrupt Mask Register, TIMER (TIMER\_DATA\_IMSK), 15-34  
 Data Length Register, RSI (RSI\_DATA\_LEN), 24-52  
 Data Timer Register, RSI (RSI\_DATA\_TMR), 24-51  
 Data Word Count Register, CRC (CRC\_DCNT), 12-33  
 Data Word Count Reload Register, CRC (CRC\_DCNTRLD), 12-34  
 Dead Time Register, PWM (PWM\_DT), 18-67  
 Debounce Register, CNT (CNT\_DEBNCE), 17-23  
 Debug Register, CAN (CAN\_DBG), 21-49  
 Debug Register, EMAC (EMAC\_DBG), 23-123  
 Delay Register, SPI (SPI\_DLY), 25-38  
 Device Control Register, USB (USB\_DEV\_CTL), 22-114  
 Divisor Register, CGU (CGU\_DIV), 3-17  
 DLL Control Register, DMC (DMC\_DLLCTL), 11-43  
 DMA Bus Mode Register, EMAC (EMAC\_DMA\_BUSMODE), 23-223  
 DMA Channel n Address Register, USB (USB\_DMAAn\_ADDR), 22-179  
 DMA Channel n Control Register, USB

(USB\_DMAAn\_CTL), 22-176  
 DMA Channel n Count Register, USB (USB\_DMAAn\_CNT), 22-179  
 DMA Controller Error interrupt, 7-9  
 DMA Interrupt Enable Register, EMAC (EMAC\_DMA\_IEN), 23-240  
 DMA Interrupt Register, USB (USB\_DMA\_IRQ), 22-173  
 DMA Missed Frame Register, EMAC (EMAC\_DMA\_MISS\_FRM), 23-243  
 DMA Operation Mode Register, EMAC (EMAC\_DMA\_OPMODE), 23-235  
 DMA Read Data Register, SDU (SDU\_DMARD), 35-24  
 DMA Rx Buffer Current Register, EMAC (EMAC\_DMA\_RXBUF\_CUR), 23-249  
 DMA Rx Descriptor Current Register, EMAC (EMAC\_DMA\_RXDSC\_CUR), 23-248  
 DMA Rx Descriptor List Address Register, EMAC (EMAC\_DMA\_RXDSC\_ADDR), 23-227  
 DMA Rx Interrupt Watch Dog Register, EMAC (EMAC\_DMA\_RXIWDOG), 23-244  
 DMA Rx Poll Demand register, EMAC (EMAC\_DMA\_RXPOLL), 23-226  
 DMA SCB Bus Mode Register, EMAC (EMAC\_DMA\_BMMODE), 23-245  
 DMA SCB Status Register, EMAC (EMAC\_DMA\_BMSTAT), 23-246  
 DMA Status Register, EMAC (EMAC\_DMA\_STAT), 23-229  
 DMA Tx Buffer Current Register, EMAC (EMAC\_DMA\_TXBUF\_CUR), 23-248  
 DMA Tx Descriptor Current Register, EMAC (EMAC\_DMA\_TXDSC\_CUR), 23-247  
 DMA Tx Descriptor List Address Register, EMAC (EMAC\_DMA\_TXDSC\_ADDR), 23-228  
 DMA Tx Poll Demand Register, EMAC (EMAC\_DMA\_TXPOLL), 23-226  
 DMA Write Data Register, SDU (SDU\_DMAWD), 35-24  
 DMA\_ADDR\_CUR (Current Address, DMA), 13-61  
 DMA\_ADDRSTART (Start Address of Current Buffer, DMA), 13-47  
 DMA\_BWLCNT (Bandwidth Limit Count, DMA), 13-68

- DMA\_BWLCNT\_CUR (Bandwidth Limit Count Current, DMA), 13-69  
 DMA\_BWMCNT (Bandwidth Monitor Count, DMA), 13-69  
 DMA\_BWMCNT\_CUR (Bandwidth Monitor Count Current, DMA), 13-70  
 DMA\_CFG (Configuration Register, DMA), 13-48  
 DMA\_DSCPTR\_CUR (Current Descriptor Pointer, DMA), 13-60  
 DMA\_DSCPTR\_NXT (Pointer to Next Initial Descriptor, DMA), 13-47  
 DMA\_DSCPTR\_PRV (Previous Initial Descriptor Pointer, DMA), 13-61  
 DMA\_STAT (Status Register, DMA), 13-62  
 DMA\_XCNT (Inner Loop Count Start Value, DMA), 13-57  
 DMA\_XCNT\_CUR (Current Count(1D) or intra-row XCNT (2D), DMA), 13-67  
 DMA\_XMOD (Inner Loop Address Increment, DMA), 13-58  
 DMA\_YCNT (Outer Loop Count Start Value (2D only), DMA), 13-58  
 DMA\_YCNT\_CUR (Current Row Count (2D only), DMA), 13-67  
 DMA\_YMOD (Outer Loop Address Increment (2D only), DMA), 13-59  
 DMC\_CFG (Configuration Register, DMC), 11-28  
 DMC\_CTL (Control Register, DMC), 11-16  
 DMC\_DLLCTL (DLL Control Register, DMC), 11-43  
 DMC\_EFFECTL (Efficiency Control Register, DMC), 11-22  
 DMC\_EMR1 (Shadow EMR1 Register, DMC), 11-38  
 DMC\_EMR2 (Shadow EMR2 Register, DMC), 11-40  
 DMC\_EMR3 (Shadow EMR3 Register, DMC), 11-42  
 DMC\_MR (Shadow MR Register, DMC), 11-35  
 DMC\_MSK (Mask (Mode Register Shadow) Register, DMC), 11-33  
 DMC\_PADCTL (PAD Control Register, DMC), 11-47  
 DMC\_PHY\_CTL1 (PHY Control 1 Register, DMC), 11-44  
 DMC\_PHY\_CTL3 (PHY Control 3 Register, DMC), 11-45  
 DMC\_PRIO (Priority ID Register, DMC), 11-26  
 DMC\_PRIOMSK (Priority ID Mask Register, DMC), 11-27  
 DMC\_STAT (Status Register, DMC), 11-19  
 DMC\_TR0 (Timing 0 Register, DMC), 11-29  
 DMC\_TR1 (Timing 1 Register, DMC), 11-31  
 DMC\_TR2 (Timing 2 Register, DMC), 11-32  
 DPM0 Event interrupt, 5-2, 7-9  
 DPM\_CCBF\_DIS (Core Clock Buffer Disable Register, DPM), 5-18  
 DPM\_CCBF\_EN (Core Clock Buffer Enable Register, DPM), 5-19  
 DPM\_CCBF\_STAT (Core Clock Buffer Status Register, DPM), 5-20  
 DPM\_CCBF\_STAT\_STKY (Core Clock Buffer Status Sticky Register, DPM), 5-20  
 DPM\_CTL (Control Register, DPM), 5-13  
 DPM\_HIB\_DIS (Hibernate Disable Register, DPM), 5-25  
 DPM\_PGCNTR (Power Good Counter Register, DPM), 5-26  
 DPM\_RESTOREn (Restore n Register, DPM), 5-28  
 DPM\_SCBF\_DIS (System Clock Buffer Disable Register, DPM), 5-21  
 DPM\_STAT (Status Register, DPM), 5-15  
 DPM\_WAKE\_EN (Wakeup Enable Register, DPM), 5-22  
 DPM\_WAKE\_POL (Wakeup Polarity Register, DPM), 5-23  
 DPM\_WAKE\_STAT (Wakeup Status Register, DPM), 5-24
- ## E
- ECC Error Address 0 Register, L2CTL (L2CTL\_ERRADDR0), 10-26  
 ECC Error Address 1 Register, L2CTL (L2CTL\_ERRADDR1), 10-26  
 ECC Error Address 2 Register, L2CTL (L2CTL\_ERRADDR2), 10-27  
 ECC Error Address 3 Register, L2CTL (L2CTL\_ERRADDR3), 10-28  
 ECC Error Address 4 Register, L2CTL (L2CTL\_ERRADDR4), 10-28  
 ECC Error Address 5 Register, L2CTL (L2CTL\_ERRADDR5), 10-29  
 ECC Error Address 6 Register, L2CTL (L2CTL\_ERRADDR6), 10-30  
 ECC Error Address 7 Register, L2CTL

- (L2CTL\_ERRADDR7), 10-30
- Efficiency Control Register, DMC (DMC\_EFFCTL), 11-22
- EMAC0 Status (Masters), 8-3
- EMAC0 Status interrupt, 7-6, 23-9
- EMAC1 Status (Masters), 8-4
- EMAC1 Status interrupt, 7-6, 23-9
- EMAC\_ADDR0\_HI (MAC Address 0 High Register, EMAC), 23-128
- EMAC\_ADDR0\_LO (MAC Address 0 Low Register, EMAC), 23-128
- EMAC\_DBG (Debug Register, EMAC), 23-123
- EMAC\_DMA\_BMMODE (DMA SCB Bus Mode Register, EMAC), 23-245
- EMAC\_DMA\_BMSTAT (DMA SCB Status Register, EMAC), 23-246
- EMAC\_DMA\_BUSMODE (DMA Bus Mode Register, EMAC), 23-223
- EMAC\_DMA\_IEN (DMA Interrupt Enable Register, EMAC), 23-240
- EMAC\_DMA\_MISS\_FRM (DMA Missed Frame Register, EMAC), 23-243
- EMAC\_DMA\_OPMODE (DMA Operation Mode Register, EMAC), 23-235
- EMAC\_DMA\_RXBUF\_CUR (DMA Rx Buffer Current Register, EMAC), 23-249
- EMAC\_DMA\_RXDSC\_ADDR (DMA Rx Descriptor List Address Register, EMAC), 23-227
- EMAC\_DMA\_RXDSC\_CUR (DMA Rx Descriptor Current Register, EMAC), 23-248
- EMAC\_DMA\_RXIWDOG (DMA Rx Interrupt Watch Dog Register, EMAC), 23-244
- EMAC\_DMA\_RXPOLL (DMA Rx Poll Demand register, EMAC), 23-226
- EMAC\_DMA\_STAT (DMA Status Register, EMAC), 23-229
- EMAC\_DMA\_TXBUF\_CUR (DMA Tx Buffer Current Register, EMAC), 23-248
- EMAC\_DMA\_TXDSC\_ADDR (DMA Tx Descriptor List Address Register, EMAC), 23-228
- EMAC\_DMA\_TXDSC\_CUR (DMA Tx Descriptor Current Register, EMAC), 23-247
- EMAC\_DMA\_TXPOLL (DMA Tx Poll Demand Register, EMAC), 23-226
- EMAC\_FLOWCTL (FLoW Control Register, EMAC), 23-120
- EMAC\_HASHTBL\_HI (Hash Table High Register, EMAC), 23-116
- EMAC\_HASHTBL\_LO (Hash Table Low Register, EMAC), 23-116
- EMAC\_IMSK (Interrupt Mask Register, EMAC), 23-127
- EMAC\_IPC\_RXIMSK (MMC IPC Rx Interrupt Mask Register, EMAC), 23-176
- EMAC\_IPC\_RXINT (MMC IPC Rx Interrupt Register, EMAC), 23-181
- EMAC\_ISTAT (Interrupt Status Register, EMAC), 23-126
- EMAC\_MACCFG (MAC Configuration Register, EMAC), 23-108
- EMAC\_MACFRMFILT (MAC Rx Frame Filter Register, EMAC), 23-113
- EMAC\_MMC\_CTL (MMC Control Register, EMAC), 23-129
- EMAC\_MMC\_RXIMSK (MMC Rx Interrupt Mask Register, EMAC), 23-139
- EMAC\_MMC\_RXINT (MMC Rx Interrupt Register, EMAC), 23-131
- EMAC\_MMC\_TXIMSK (MMC TX Interrupt Mask Register, EMAC), 23-143
- EMAC\_MMC\_TXINT (MMC Tx Interrupt Register, EMAC), 23-135
- EMAC\_RX1024TOMAX\_GB (Rx 1024- to Max-Byte Frames (Good/Bad) Register, EMAC), 23-171
- EMAC\_RX128TO255\_GB (Rx 128- to 255-Byte Frames (Good/Bad) Register, EMAC), 23-169
- EMAC\_RX256TO511\_GB (Rx 256- to 511-Byte Frames (Good/Bad) Register, EMAC), 23-170
- EMAC\_RX512TO1023\_GB (Rx 512- to 1023-Byte Frames (Good/Bad) Register, EMAC), 23-170
- EMAC\_RX64\_GB (Rx 64-Byte Frames (Good/Bad) Register, EMAC), 23-168
- EMAC\_RX65TO127\_GB (Rx 65- to 127-Byte Frames (Good/Bad) Register, EMAC), 23-168
- EMAC\_RXALIGN\_ERR (Rx alignment Error Register, EMAC), 23-165
- EMAC\_RXBCASTFRM\_G (Rx Broadcast Frames (Good) Register, EMAC), 23-163
- EMAC\_RXCRC\_ERR (Rx CRC Error Register, EMAC), 23-164

- EMAC\_RXFIFO\_OVF (Rx FIFO Overflow Register, EMAC), 23-174
- EMAC\_RXFRMCNT\_GB (Rx Frame Count (Good/Bad) Register, EMAC), 23-161
- EMAC\_RXICMP\_ERR\_FRM (Rx ICMP Error Frames Register, EMAC), 23-193
- EMAC\_RXICMP\_ERR\_OCT (Rx ICMP Error Octets Register, EMAC), 23-202
- EMAC\_RXICMP\_GD\_FRM (Rx ICMP Good Frames Register, EMAC), 23-193
- EMAC\_RXICMP\_GD\_OCT (Rx ICMP Good Octets Register, EMAC), 23-201
- EMAC\_RXIPV4\_FRAG\_FRM (Rx IPv4 Datagrams Fragmented Frames Register, EMAC), 23-187
- EMAC\_RXIPV4\_FRAG\_OCT (Rx IPv4 Datagrams Fragmented Octets Register, EMAC), 23-196
- EMAC\_RXIPV4\_GD\_FRM (Rx IPv4 Datagrams (Good) Register, EMAC), 23-185
- EMAC\_RXIPV4\_GD\_OCT (Rx IPv4 Datagrams Good Octets Register, EMAC), 23-194
- EMAC\_RXIPV4\_HDR\_ERR\_FRM (Rx IPv4 Datagrams Header Errors Register, EMAC), 23-186
- EMAC\_RXIPV4\_HDR\_ERR\_OCT (Rx IPv4 Datagrams Header Errors Register, EMAC), 23-194
- EMAC\_RXIPV4\_NOPAY\_FRM (Rx IPv4 Datagrams No Payload Frame Register, EMAC), 23-187
- EMAC\_RXIPV4\_NOPAY\_OCT (Rx IPv4 Datagrams No Payload Octets Register, EMAC), 23-195
- EMAC\_RXIPV4\_UDSBL\_FRM (Rx IPv4 UDP Disabled Frames Register, EMAC), 23-188
- EMAC\_RXIPV4\_UDSBL\_OCT (Rx IPv4 UDP Disabled Octets Register, EMAC), 23-196
- EMAC\_RXIPV6\_GD\_FRM (Rx IPv6 Datagrams Good Frames Register, EMAC), 23-188
- EMAC\_RXIPV6\_GD\_OCT (Rx IPv6 Good Octets Register, EMAC), 23-197
- EMAC\_RXIPV6\_HDR\_ERR\_FRM (Rx IPv6 Datagrams Header Error Frames Register, EMAC), 23-189
- EMAC\_RXIPV6\_HDR\_ERR\_OCT (Rx IPv6 Header Errors Register, EMAC), 23-197
- EMAC\_RXIPV6\_NOPAY\_FRM (Rx IPv6 Datagrams No Payload Frames Register, EMAC), 23-190
- EMAC\_RXIPV6\_NOPAY\_OCT (Rx IPv6 No Payload Octets Register, EMAC), 23-198
- EMAC\_RXJAB\_ERR (Rx Jab Error Register, EMAC), 23-166
- EMAC\_RXLEN\_ERR (Rx Length Error Register, EMAC), 23-172
- EMAC\_RXMCASTFRM\_G (Rx Multicast Frames (Good) Register, EMAC), 23-164
- EMAC\_RXOCTCNT\_G (Rx Octet Count (Good) Register, EMAC), 23-162
- EMAC\_RXOCTCNT\_GB (Rx Octet Count (Good/Bad) Register, EMAC), 23-162
- EMAC\_RXOORTYPE (Rx Out Of Range Type Register, EMAC), 23-173
- EMAC\_RXOSIZE\_G (Rx Oversize (Good) Register, EMAC), 23-167
- EMAC\_RXPAUSEFRM (Rx Pause Frames Register, EMAC), 23-173
- EMAC\_RXRUNT\_ERR (Rx Runt Error Register, EMAC), 23-165
- EMAC\_RXTCP\_ERR\_FRM (Rx TCP Error Frames Register, EMAC), 23-192
- EMAC\_RXTCP\_ERR\_OCT (Rx TCP Error Octets Register, EMAC), 23-200
- EMAC\_RXTCP\_GD\_FRM (Rx TCP Good Frames Register, EMAC), 23-191
- EMAC\_RXTCP\_GD\_OCT (Rx TCP Good Octets Register, EMAC), 23-200
- EMAC\_RXUCASTFRM\_G (Rx Unicast Frames (Good) Register, EMAC), 23-171
- EMAC\_RXUDP\_ERR\_FRM (Rx UDP Error Frames Register, EMAC), 23-191
- EMAC\_RXUDP\_ERR\_OCT (Rx UDP Error Octets Register, EMAC), 23-199
- EMAC\_RXUDP\_GD\_FRM (Rx UDP Good Frames Register, EMAC), 23-190
- EMAC\_RXUDP\_GD\_OCT (Rx UDP Good Octets Register, EMAC), 23-199
- EMAC\_RXUSIZE\_G (Rx Undersize (Good) Register, EMAC), 23-167
- EMAC\_RXVLANFRM\_GB (Rx VLAN Frames (Good/Bad) Register, EMAC), 23-174
- EMAC\_RXWDOG\_ERR (Rx Watch Dog Error Register, EMAC), 23-175
- EMAC\_SMI\_ADDR (SMI Address Register, EMAC), 23-117
- EMAC\_SMI\_DATA (SMI Data Register, EMAC), 23-119

- EMAC\_TM\_ADDEND (Time Stamp Addend Register, EMAC), 23-212
- EMAC\_TM\_AUXSTMP\_NSEC (Time Stamp Auxiliary TS Nano Seconds Register, EMAC), 23-220
- EMAC\_TM\_AUXSTMP\_SEC (Time Stamp Auxiliary TM Seconds Register, EMAC), 23-220
- EMAC\_TM\_CTL (Time Stamp Control Register, EMAC), 23-203
- EMAC\_TM\_HISEC (Time Stamp High Second Register, EMAC), 23-214
- EMAC\_TM\_NSEC (Time Stamp Nano Seconds Register, EMAC), 23-209
- EMAC\_TM\_NSECUPDT (Time Stamp Nano Seconds Update Register, EMAC), 23-211
- EMAC\_TM\_NTGTM (Time Stamp Target Time Nano Seconds Register, EMAC), 23-213
- EMAC\_TM\_PPSCTL (PPS Control Register, EMAC), 23-217
- EMAC\_TM\_PPSINTVL (Time Stamp PPS Interval Register, EMAC), 23-221
- EMAC\_TM\_PPSWIDTH (PPS Width Register, EMAC), 23-222
- EMAC\_TM\_SEC (Time Stamp Low Seconds Register, EMAC), 23-209
- EMAC\_TM\_SECUPDT (Time Stamp Seconds Update Register, EMAC), 23-210
- EMAC\_TM\_STMPSTAT (Time Stamp Status Register, EMAC), 23-215
- EMAC\_TM\_SUBSEC (Time Stamp Sub Second Increment Register, EMAC), 23-208
- EMAC\_TM\_TGTM (Time Stamp Target Time Seconds Register, EMAC), 23-212
- EMAC\_TX1024TOMAX\_GB (Tx 1024- to Max-Byte Frames (Good/Bad) Register, EMAC), 23-152
- EMAC\_TX128TO255\_GB (Tx 128- to 255-Byte Frames (Good/Bad) Register, EMAC), 23-150
- EMAC\_TX256TO511\_GB (Tx 256- to 511-Byte Frames (Good/Bad) Register, EMAC), 23-150
- EMAC\_TX512TO1023\_GB (Tx 512- to 1023-Byte Frames (Good/Bad) Register, EMAC), 23-151
- EMAC\_TX64\_GB (Tx 64-Byte Frames (Good/Bad) Register, EMAC), 23-149
- EMAC\_TX65TO127\_GB (Tx 65- to 127-Byte Frames (Good/Bad) Register, EMAC), 23-149
- EMAC\_TXBCASTFRM\_G (Tx Broadcast Frames (Good) Register, EMAC), 23-147
- EMAC\_TXBCASTFRM\_GB (Tx Broadcast Frames (Good/Bad) Register, EMAC), 23-153
- EMAC\_TXCARR\_ERR (Tx Carrier Error Register, EMAC), 23-158
- EMAC\_TXDEFERRED (Tx Deferred Register, EMAC), 23-156
- EMAC\_TXEXCESSCOL (Tx Excess Collision Register, EMAC), 23-157
- EMAC\_TXEXCESSDEF (Tx Excess Deferral Register, EMAC), 23-159
- EMAC\_TXFRMCNT\_G (Tx Frame Count (Good) Register, EMAC), 23-159
- EMAC\_TXFRMCNT\_GB (Tx Frame Count (Good/Bad) Register, EMAC), 23-147
- EMAC\_TXLATECOL (Tx Late Collision Register, EMAC), 23-156
- EMAC\_TXMCASTFRM\_G (Tx Multicast Frames (Good) Register, EMAC), 23-148
- EMAC\_TXMCASTFRM\_GB (Tx Multicast Frames (Good/Bad) Register, EMAC), 23-153
- EMAC\_TXMULTCOL\_G (Tx Multiple Collision (Good) Register, EMAC), 23-155
- EMAC\_TXOCTCNT\_G (Tx Octet Count (Good) Register, EMAC), 23-158
- EMAC\_TXOCTCNT\_GB (Tx OCT Count (Good/Bad) Register, EMAC), 23-146
- EMAC\_TXPAUSEFRM (Tx Pause Frame Register, EMAC), 23-160
- EMAC\_TXSNGCOL\_G (Tx Single Collision (Good) Register, EMAC), 23-155
- EMAC\_TXUCASTFRM\_GB (Tx Unicast Frames (Good/Bad) Register, EMAC), 23-152
- EMAC\_TXUNDR\_ERR (Tx Underflow Error Register, EMAC), 23-154
- EMAC\_TXVLANFRM\_G (Tx VLAN Frames (Good) Register, EMAC), 23-161
- EMAC\_VLANTAG (VLAN Tag Register, EMAC), 23-122
- Endpoint Information Register, USB (USB\_EPINFO), 22-122
- EP0 Configuration and Status (Host) Register, USB (USB\_EP0\_CSRn\_H), 22-133
- EP0 Configuration and Status (Peripheral) Register, USB (USB\_EP0\_CSRn\_P), 22-137

- EP0 Configuration Information Register, USB (USB\_EP0\_CFGDATAn), 22-171
- EP0 Connection Type Register, USB (USB\_EP0\_TYPEn), 22-165
- EP0 NAK Limit Register, USB (USB\_EP0\_NAKLIMITn), 22-165
- EP0 Number of Received Bytes Register, USB (USB\_EP0\_CNTn), 22-161
- EPn Number of Bytes Received Register, USB (USB\_EPn\_RXCNT), 22-161
- EPn Receive Configuration and Status (Host) Register, USB (USB\_EPn\_RXCSR\_H), 22-150
- EPn Receive Configuration and Status (Peripheral) Register, USB (USB\_EPn\_RXCSR\_P), 22-156
- EPn Receive Maximum Packet Length Register, USB (USB\_EPn\_RXMAXP), 22-149
- EPn Receive Polling Interval Register, USB (USB\_EPn\_RXINTERVAL), 22-170
- EPn Receive Type Register, USB (USB\_EPn\_RXTYPE), 22-167
- EPn Request Packet Count Register, USB (USB\_RQPKTCNTn), 22-180
- EPn Transmit Configuration and Status (Host) Register, USB (USB\_EPn\_TXCSR\_H), 22-140
- EPn Transmit Configuration and Status (Peripheral) Register, USB (USB\_EPn\_TXCSR\_P), 22-145
- EPn Transmit Maximum Packet Length Register, USB (USB\_EPn\_TXMAXP), 22-132
- EPn Transmit Polling Interval Register, USB (USB\_EPn\_TXINTERVAL), 22-166
- EPn Transmit Type Register, USB (USB\_EPn\_TXTYPE), 22-162
- EPPI0 Channel 0 DMA (Masters), 8-5, 31-7
- EPPI0 Channel 0 DMA (Slaves), 8-9, 31-7
- EPPI0 Channel 0 DMA interrupt, 7-8, 31-6
- EPPI0 Channel 1 DMA (Masters), 8-5, 31-7
- EPPI0 Channel 1 DMA (Slaves), 8-9, 31-8
- EPPI0 Channel 1 DMA interrupt, 7-8, 31-6
- EPPI0 Status interrupt, 7-8, 31-7
- EPPI1 Channel 0 DMA (Masters), 8-5, 31-7
- EPPI1 Channel 0 DMA (Slaves), 8-9, 31-8
- EPPI1 Channel 0 DMA interrupt, 7-8, 31-7
- EPPI1 Channel 1 DMA (Masters), 8-5, 31-7
- EPPI1 Channel 1 DMA (Slaves), 8-9, 31-8
- EPPI1 Channel 1 DMA interrupt, 7-8, 31-7
- EPPI1 Status interrupt, 7-8, 31-7
- EPPI2 Channel 0 DMA (Masters), 8-5, 31-7
- EPPI2 Channel 0 DMA (Slaves), 8-9, 31-8
- EPPI2 Channel 0 DMA interrupt, 7-8, 31-7
- EPPI2 Channel 1 DMA (Masters), 8-5, 31-7
- EPPI2 Channel 1 DMA (Slaves), 8-9, 31-8
- EPPI2 Channel 1 DMA interrupt, 7-8, 31-7
- EPPI2 Status interrupt, 7-8, 31-7
- EPPI\_CLKDIV (Clock Divide Register, EPPI), 31-76
- EPPI\_CTL (Control Register, EPPI), 31-77
- EPPI\_CTL2 (Control Register 2, EPPI), 31-96
- EPPI\_EVENCLIP (Clipping Register for EVEN (Luma) Data, EPPI), 31-94
- EPPI\_FRAME (Lines Per Frame Register, EPPI), 31-74
- EPPI\_FS1\_DLY (Frame Sync 1 Delay Value, EPPI), 31-94
- EPPI\_FS1\_PASPL (FS1 Period Register / EPPI Active Samples Per Line Register, EPPI), 31-88
- EPPI\_FS1\_WLHB (FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register, EPPI), 31-87
- EPPI\_FS2\_DLY (Frame Sync 2 Delay Value, EPPI), 31-95
- EPPI\_FS2\_PALPF (FS2 Period Register / EPPI Active Lines Per Field Register, EPPI), 31-90
- EPPI\_FS2\_WLVB (FS2 Width Register / EPPI Lines Of Vertical Blanking Register, EPPI), 31-89
- EPPI\_HCNT (Horizontal Transfer Count Register, EPPI), 31-72
- EPPI\_HDLY (Horizontal Delay Count Register, EPPI), 31-72
- EPPI\_IMSK (Interrupt Mask Register, EPPI), 31-91
- EPPI\_LINE (Samples Per Line Register, EPPI), 31-75
- EPPI\_ODDCLIP (Clipping Register for ODD (Chroma) Data, EPPI), 31-93
- EPPI\_STAT (Status Register, EPPI), 31-69
- EPPI\_VCNT (Vertical Transfer Count Register, EPPI), 31-73
- EPPI\_VDLY (Vertical Delay Count Register, EPPI), 31-74
- Error Address Register, TRU (TRU\_ERRADDR), 8-16
- Error Counter Register, CAN (CAN\_CEC), 21-54
- Error Counter Warning Level Register, CAN

(CAN\_EWR), 21-68  
 Error Status Register, CAN (CAN\_ESR), 21-69  
 Error Type 0 Address Register, L2CTL (L2CTL\_EADDR0), 10-33  
 Error Type 0 Register, L2CTL (L2CTL\_ET0), 10-32  
 Error Type 1 Address Register, L2CTL (L2CTL\_EADDR1), 10-35  
 Error Type 1 Register, L2CTL (L2CTL\_ET1), 10-34  
 Error Type Status Register, TIMER (TIMER\_ERR\_TYPE), 15-38  
 Exception Mask Register, RSI (RSI\_IMSK0), 24-67  
 Exception Status Register, RSI (RSI\_STAT0), 24-65

## F

Fault Control Register, SEC (SEC\_FCTL), 7-28  
 Fault COP Period Current Register, SEC (SEC\_FCOPP\_CUR), 7-39  
 Fault COP Period Register, SEC (SEC\_FCOPP), 7-38  
 Fault Delay Current Register, SEC (SEC\_FDLY\_CUR), 7-36  
 Fault Delay Register, SEC (SEC\_FDLY), 7-36  
 Fault End Register, SEC (SEC\_FEND), 7-35  
 Fault Source ID Register, SEC (SEC\_FSID), 7-34  
 Fault Status Register, SEC (SEC\_FSTAT), 7-31  
 Fault System Reset Delay Current Register, SEC (SEC\_FSRDLY\_CUR), 7-38  
 Fault System Reset Delay Register, SEC (SEC\_FSRDLY), 7-37  
 FIFO Byte (8-Bit) Register, USB (USB\_FIFOBn), 22-112  
 FIFO Control Register, TWI (TWI\_FIFOCTL), 20-37  
 FIFO Counter Register, RSI (RSI\_FIFO\_CNT), 24-61  
 FIFO Half-Word (16-Bit) Register, USB (USB\_FIFOHn), 22-113  
 FIFO Status Register, TWI (TWI\_FIFOSTAT), 20-38  
 FIFO Word (32-Bit) Register, USB (USB\_FIFOn), 22-114  
 Fill Value Register, CRC (CRC\_FILLVAL), 12-35  
 FLow Control Register, EMAC (EMAC\_FLOWCTL), 23-120  
 Frame Number Register, USB (USB\_FRAME), 22-109  
 Frame Sync 1 Delay Value, EPPI (EPPI\_FS1\_DLY), 31-94  
 Frame Sync 2 Delay Value, EPPI (EPPI\_FS2\_DLY),

31-95  
 FS1 Period Register / EPPI Active Samples Per Line Register, EPPI (EPPI\_FS1\_PASPL), 31-88  
 FS1 Width Register / EPPI Horizontal Blanking Samples Per Line Register, EPPI (EPPI\_FS1\_WLHB), 31-87  
 FS2 Period Register / EPPI Active Lines Per Field Register, EPPI (EPPI\_FS2\_PALPF), 31-90  
 FS2 Width Register / EPPI Lines Of Vertical Blanking Register, EPPI (EPPI\_FS2\_WLVB), 31-89  
 Full-Speed EOF 1 Register, USB (USB\_FS\_EOF1), 22-125  
 Function Address Register, USB (USB\_FADDR), 22-89

## G

Global CAN Interrupt Flag Register, CAN (CAN\_GIF), 21-61  
 Global CAN Interrupt Mask Register, CAN (CAN\_GIM), 21-58  
 Global CAN Interrupt Status Register, CAN (CAN\_GIS), 21-55  
 Global Control Register, SEC (SEC\_GCTL), 7-40  
 Global Control Register, SWU (SWU\_GCTL), 36-10  
 Global Control Register, TRU (TRU\_GCTL), 8-18  
 Global End Register, SEC (SEC\_END), 7-43  
 Global Raise Register, SEC (SEC\_RAISE), 7-43  
 Global Status Register, SEC (SEC\_GSTAT), 7-41  
 Global Status Register, SWU (SWU\_GSTAT), 36-12  
 Grant Control Register, SMC (SMC\_GCTL), 9-24  
 Grant Status Register, SMC (SMC\_GSTAT), 9-25  
 Group Halt Register, SDU (SDU\_GHLT), 35-28  
 GU Conversion Component Register, PIXC (PIXC\_CONGU), 32-47

## H

Half SPORT 'A' Control 2 Register, SPORT (SPORT\_CTL2\_A), 26-71  
 Half SPORT 'A' Control Register, SPORT (SPORT\_CTL\_A), 26-52  
 Half SPORT 'A' Divisor Register, SPORT (SPORT\_DIV\_A), 26-62  
 Half SPORT 'A' Error Register, SPORT (SPORT\_ERR\_A), 26-68



Half SPORT 'A' Multi-channel 0-31 Select Register, SPORT (SPORT\_CS0\_A), 26-65

Half SPORT 'A' Multi-channel 32-63 Select Register, SPORT (SPORT\_CS1\_A), 26-66

Half SPORT 'A' Multi-channel 64-95 Select Register, SPORT (SPORT\_CS2\_A), 26-67

Half SPORT 'A' Multi-channel 96-127 Select Register, SPORT (SPORT\_CS3\_A), 26-67

Half SPORT 'A' Multi-channel Control Register, SPORT (SPORT\_MCTL\_A), 26-63

Half SPORT 'A' Multi-channel Status Register, SPORT (SPORT\_MSTAT\_A), 26-71

Half SPORT 'A' Rx Buffer (Primary) Register, SPORT (SPORT\_RXPRI\_A), 26-74

Half SPORT 'A' Rx Buffer (Secondary) Register, SPORT (SPORT\_RXSEC\_A), 26-76

Half SPORT 'A' Tx Buffer (Primary) Register, SPORT (SPORT\_TXPRI\_A), 26-73

Half SPORT 'A' Tx Buffer (Secondary) Register, SPORT (SPORT\_TXSEC\_A), 26-75

Half SPORT 'B' Control 2 Register, SPORT (SPORT\_CTL2\_B), 26-96

Half SPORT 'B' Control Register, SPORT (SPORT\_CTL\_B), 26-77

Half SPORT 'B' Divisor Register, SPORT (SPORT\_DIV\_B), 26-87

Half SPORT 'B' Error Register, SPORT (SPORT\_ERR\_B), 26-93

Half SPORT 'B' Multi-channel 0-31 Select Register, SPORT (SPORT\_CS0\_B), 26-90

Half SPORT 'B' Multi-channel 32-63 Select Register, SPORT (SPORT\_CS1\_B), 26-91

Half SPORT 'B' Multichannel 64-95 Select Register, SPORT (SPORT\_CS2\_B), 26-92

Half SPORT 'B' Multichannel 96-127 Select Register, SPORT (SPORT\_CS3\_B), 26-92

Half SPORT 'B' Multi-channel Control Register, SPORT (SPORT\_MCTL\_B), 26-88

Half SPORT 'B' Multi-channel Status Register, SPORT (SPORT\_MSTAT\_B), 26-96

Half SPORT 'B' Rx Buffer (Primary) Register, SPORT (SPORT\_RXPRI\_B), 26-99

Half SPORT 'B' Rx Buffer (Secondary) Register, SPORT (SPORT\_RXSEC\_B), 26-101

Half SPORT 'B' Tx Buffer (Primary) Register, SPORT

(SPORT\_TXPRI\_B), 26-98

Half SPORT 'B' Tx Buffer (Secondary) Register, SPORT (SPORT\_TXSEC\_B), 26-100

Hash Table High Register, EMAC (EMAC\_HASHTBL\_HI), 23-116

Hash Table Low Register, EMAC (EMAC\_HASHTBL\_LO), 23-116

Hibernate Disable Register, DPM (DPM\_HIB\_DIS), 5-25

High Speed Timeout Register, USB (USB\_CT\_HSBT), 22-182

High-Speed EOF 1 Register, USB (USB\_HS\_EOF1), 22-125

Horizontal Delay Count Register, EPPI (EPPI\_HDL\_Y), 31-72

Horizontal Transfer Count Register, EPPI (EPPI\_HCNT), 31-72

Host High Speed Return to Normal Register, USB (USB\_CT\_HHSRTN), 22-181

## I

ID Code Register, SDU (SDU\_IDCODE), 35-13

ID Register n, SWU (SWU\_IDn), 36-23

IIMn Configuration, PVP (PVP\_IIMn\_CFG), 30-146

IIMn Control, PVP (PVP\_IIMn\_CTL), 30-148

IIMn Scaling Values, PVP (PVP\_IIMn\_SCALE), 30-149

IIMn Signed Overflow Status, PVP (PVP\_IIMn\_SOVF\_STAT), 30-150

IIMn Unsigned Overflow Status, PVP (PVP\_IIMn\_UOVF\_STAT), 30-151

Index Register, USB (USB\_INDEX), 22-110

Inner Loop Address Increment, DMA (DMA\_XMOD), 13-58

Inner Loop Count Start Value, DMA (DMA\_XCNT), 13-57

Interrupt 0 Mask Register, RSI (RSI\_XFR\_IMSK0), 24-57

Interrupt 1 Mask Register, RSI (RSI\_XFR\_IMSK1), 24-59

Interrupt Enable Clear Register, CRC (CRC\_INEN\_CLR), 12-38

Interrupt Enable Register, CRC (CRC\_INEN), 12-36

Interrupt Enable Set Register, CRC (CRC\_INEN\_SET), 12-37

Interrupt Latch Register, PWM (PWM\_ILAT), 18-64  
 Interrupt Latch Status n, PVP (PVP\_ILAT), 30-125  
 Interrupt Mask Clear Register, SPI (SPI\_IMSK\_CLR), 25-49  
 Interrupt Mask Clear Register, UART (UART\_IMSK\_CLR), 19-47  
 Interrupt Mask n, PVP (PVP\_IMSKn), 30-113  
 Interrupt Mask Register, CNT (CNT\_IMSK), 17-18  
 Interrupt Mask Register, EMAC (EMAC\_IMSK), 23-127  
 Interrupt Mask Register, EPPI (EPPI\_IMSK), 31-91  
 Interrupt Mask Register, PWM (PWM\_IMSK), 18-62  
 Interrupt Mask Register, SPI (SPI\_IMSK), 25-47  
 Interrupt Mask Register, TWI (TWI\_IMSK), 20-35  
 Interrupt Mask Register, UART (UART\_IMSK), 19-41  
 Interrupt Mask Set Register, SPI (SPI\_IMSK\_SET), 25-50  
 Interrupt Mask Set Register, UART (UART\_IMSK\_SET), 19-45  
 Interrupt Pending Register, CAN (CAN\_INT), 21-66  
 Interrupt Request n, PVP (PVP\_IREQn), 30-130  
 Interrupt Status Register, EMAC (EMAC\_ISTAT), 23-126  
 Interrupt Status Register, PIXC (PIXC\_IRQSTAT), 32-45  
 Interrupt Status Register, TWI (TWI\_ISTAT), 20-31  
 IPF0 (Camera Pipe) Configuration, PVP (PVP\_IPF0\_CFG), 30-164  
 IPF0 (Camera Pipe) Horizontal Position, PVP (PVP\_IPF0\_HPOS), 30-172  
 IPF0 (Camera Pipe) Vertical Position, PVP (PVP\_IPF0\_VPOS), 30-173  
 IPF1 (Memory Pipe) Configuration, PVP (PVP\_IPF1\_CFG), 30-174  
 IPFn (Camera/Memory Pipe) Control, PVP (PVP\_IPFn\_CTL), 30-166  
 IPFn (Camera/Memory Pipe) Frame Count, PVP (PVP\_IPFn\_FCNT), 30-170  
 IPFn (Camera/Memory Pipe) Horizontal Count, PVP (PVP\_IPFn\_HCNT), 30-171  
 IPFn (Camera/Memory Pipe) Pipe Control, PVP (PVP\_IPFn\_PIPECTL), 30-165  
 IPFn (Camera/Memory Pipe) TAG Status, PVP (PVP\_IPFn\_TAG\_STAT), 30-173

IPFn (Camera/Memory Pipe) TAG Value, PVP (PVP\_IPFn\_TAG), 30-170  
 IPFn (Camera/Memory Pipe) Vertical Count, PVP (PVP\_IPFn\_VCNT), 30-171

## L

L2CTL0 ECC Error interrupt, 7-3, 10-2  
 L2CTL\_ACTL\_C0 (Access Control Core 0 Register, L2CTL), 10-12  
 L2CTL\_ACTL\_C1 (Access Control Core 1 Register, L2CTL), 10-15  
 L2CTL\_ACTL\_SYS (Access Control System Register, L2CTL), 10-17  
 L2CTL\_CTL (Control Register, L2CTL), 10-8  
 L2CTL\_EADDR0 (Error Type 0 Address Register, L2CTL), 10-33  
 L2CTL\_EADDR1 (Error Type 1 Address Register, L2CTL), 10-35  
 L2CTL\_ERRADDR0 (ECC Error Address 0 Register, L2CTL), 10-26  
 L2CTL\_ERRADDR1 (ECC Error Address 1 Register, L2CTL), 10-26  
 L2CTL\_ERRADDR2 (ECC Error Address 2 Register, L2CTL), 10-27  
 L2CTL\_ERRADDR3 (ECC Error Address 3 Register, L2CTL), 10-28  
 L2CTL\_ERRADDR4 (ECC Error Address 4 Register, L2CTL), 10-28  
 L2CTL\_ERRADDR5 (ECC Error Address 5 Register, L2CTL), 10-29  
 L2CTL\_ERRADDR6 (ECC Error Address 6 Register, L2CTL), 10-30  
 L2CTL\_ERRADDR7 (ECC Error Address 7 Register, L2CTL), 10-30  
 L2CTL\_ET0 (Error Type 0 Register, L2CTL), 10-32  
 L2CTL\_ET1 (Error Type 1 Register, L2CTL), 10-34  
 L2CTL\_RFA (Refresh Address Register, L2CTL), 10-25  
 L2CTL\_RPCR (Read Priority Count Register, L2CTL), 10-23  
 L2CTL\_STAT (Status Register, L2CTL), 10-20  
 L2CTL\_WPCR (Write Priority Count Register, L2CTL), 10-24  
 Line Per Frame Register, PIXC (PIXC\_LPF), 32-40  
 Lines Per Frame Register, EPPI (EPPI\_FRAME),

31-74

Link Information Register, USB (USB\_LINKINFO), 22-123

Lower Address Register n, SWU (SWU\_LAn), 36-21

Low-Speed EOF 1 Register, USB (USB\_LS\_EOF1), 22-126

LP0 DMA Channel (Masters), 8-4, 28-3

LP0 DMA Channel (Slaves), 8-8, 28-3

LP0 DMA Channel interrupt, 7-6, 28-2

LP0 Status interrupt, 7-6, 28-2

LP1 DMA Channel (Masters), 8-4, 28-3

LP1 DMA Channel (Slaves), 8-8, 28-3

LP1 DMA Channel interrupt, 7-6, 28-2

LP1 Status interrupt, 7-6, 28-2

LP2 DMA Channel (Masters), 8-4, 28-3

LP2 DMA Channel (Slaves), 8-8, 28-3

LP2 DMA Channel interrupt, 7-6, 28-2

LP2 Status interrupt, 7-6, 28-2

LP3 DMA Channel (Masters), 8-4, 28-3

LP3 DMA Channel (Slaves), 8-8, 28-3

LP3 DMA Channel interrupt, 7-6, 28-2

LP3 Status interrupt, 7-6, 28-3

LP\_CTL (Control Register, LP), 28-19

LP\_DIV (Clock Divider Value, LP), 28-23

LPM Attribute Register, USB (USB\_LPM\_ATTR), 22-183

LPM Control Register, USB (USB\_LPM\_CTL), 22-184

LPM Function Address Register, USB (USB\_LPM\_FADDR), 22-191

LPM Interrupt Enable Register, USB (USB\_LPM\_IEN), 22-187

LPM Interrupt Status Register, USB (USB\_LPM\_IRQ), 22-188

LP\_RX (Receive Buffer, LP), 28-24

LP\_STAT (Status Register, LP), 28-21

LP\_TX (Transmit Buffer, LP), 28-24

LP\_TXIN\_SHDW (Shadow Input Transmit Buffer, LP), 28-25

LP\_TXOUT\_SHDW (Shadow Output Transmit Buffer, LP), 28-26

## M

MAC Address 0 High Register, EMAC (EMAC\_ADDR0\_HI), 23-128

MAC Address 0 Low Register, EMAC (EMAC\_ADDR0\_LO), 23-128

MAC Configuration Register, EMAC (EMAC\_MACCFG), 23-108

MAC Rx Frame Filter Register, EMAC (EMAC\_MACFRMFILT), 23-113

Mailbox Configuration 1 Register, CAN (CAN\_MC1), 21-32

Mailbox Configuration 2 Register, CAN (CAN\_MC2), 21-39

Mailbox Direction 1 Register, CAN (CAN\_MD1), 21-32

Mailbox Direction 2 Register, CAN (CAN\_MD2), 21-40

Mailbox ID 0 Register, CAN (CAN\_MBnn\_ID0), 21-81

Mailbox ID 1 Register, CAN (CAN\_MBnn\_ID1), 21-82

Mailbox Interrupt Mask 1 Register, CAN (CAN\_MBIM1), 21-37

Mailbox Interrupt Mask 2 Register, CAN (CAN\_MBIM2), 21-45

Mailbox Length Register, CAN (CAN\_MBnn\_LENGTH), 21-80

Mailbox Receive Interrupt Flag 1 Register, CAN (CAN\_MBRIF1), 21-37

Mailbox Receive Interrupt Flag 2 Register, CAN (CAN\_MBRIF2), 21-44

Mailbox Timestamp Register, CAN (CAN\_MBnn\_TIMESTAMP), 21-80

Mailbox Transmit Interrupt Flag 1 Register, CAN (CAN\_MBTIF1), 21-36

Mailbox Transmit Interrupt Flag 2 Register, CAN (CAN\_MBTIF2), 21-43

Mailbox Word 0 Register, CAN (CAN\_MBnn\_DATA0), 21-77

Mailbox Word 1 Register, CAN (CAN\_MBnn\_DATA1), 21-78

Mailbox Word 2 Register, CAN (CAN\_MBnn\_DATA2), 21-79

Mailbox Word 3 Register, CAN (CAN\_MBnn\_DATA3), 21-79

Mask (Mode Register Shadow) Register, DMC (DMC\_MSK), 11-33

Masked Interrupt Clear Register, SPI

- (SPI\_ILAT\_CLR), 25-59
- Masked Interrupt Condition Register, SPI (SPI\_ILAT), 25-57
- Master Interfaces Number Register, SCB (SCB\_MASTERS), 2-24
- Master Mode Address Register, TWI (TWI\_MSTRADDR), 20-31
- Master Mode Control Registers, TWI (TWI\_MSTRCTL), 20-24
- Master Mode Status Register, TWI (TWI\_MSTRSTAT), 20-27
- Master Trigger Register, TRU (TRU\_MTR), 8-15
- Maximum Count Register, CNT (CNT\_MAX), 17-27
- maximum individual packet size (MaxPktSize); MaxPktSize (maximum individual packet size), 22-28, 22-29, 22-30, 22-31, 22-32
- Memory Access Address Register, SDU (SDU\_MACADDR), 35-23
- Memory Access Control Register, SDU (SDU\_MACCTL), 35-21
- Memory Access Data Register, SDU (SDU\_MACDATA), 35-23
- Memory DMA Stream 0 Destination / CRC0 Output Channel (Masters), 8-4
- Memory DMA Stream 0 Destination / CRC0 Output Channel (Slaves), 8-9
- Memory DMA Stream 0 Destination / CRC0 Output Channel interrupt, 7-7
- Memory DMA Stream 0 Source / CRC0 Input Channel (Masters), 8-4
- Memory DMA Stream 0 Source / CRC0 Input Channel (Slaves), 8-8
- Memory DMA Stream 0 Source / CRC0 Input Channel interrupt, 7-7
- Memory DMA Stream 1 Destination / CRC1 Output Channel (Masters), 8-4
- Memory DMA Stream 1 Destination / CRC1 Output Channel (Slaves), 8-9
- Memory DMA Stream 1 Destination / CRC1 Output Channel interrupt, 7-7
- Memory DMA Stream 1 Source / CRC1 Input Channel (Masters), 8-4
- Memory DMA Stream 1 Source / CRC1 Input Channel (Slaves), 8-9
- Memory DMA Stream 1 Source / CRC1 Input Channel interrupt, 7-7
- Memory DMA Stream 2 Destination Channel (Masters), 8-4
- Memory DMA Stream 2 Destination Channel (Slaves), 8-9
- Memory DMA Stream 2 Destination Channel interrupt, 7-7
- Memory DMA Stream 2 Source Channel (Masters), 8-4
- Memory DMA Stream 2 Source Channel (Slaves), 8-9
- Memory DMA Stream 2 Source Channel interrupt, 7-7
- Memory DMA Stream 3 Destination Channel (Masters), 8-4
- Memory DMA Stream 3 Destination Channel (Slaves), 8-9
- Memory DMA Stream 3 Destination Channel interrupt, 7-8
- Memory DMA Stream 3 Source Channel (Masters), 8-4
- Memory DMA Stream 3 Source Channel (Slaves), 8-9
- Memory DMA Stream 3 Source Channel interrupt, 7-8
- Message Clear Register, SDU (SDU\_MSG\_CLR), 35-27
- Message Register, SDU (SDU\_MSG), 35-25
- Message Set Register, SDU (SDU\_MSG\_SET), 35-26
- Minimum Count Register, CNT (CNT\_MIN), 17-27
- MMC Control Register, EMAC (EMAC\_MMC\_CTL), 23-129
- MMC IPC Rx Interrupt Mask Register, EMAC (EMAC\_IPC\_RXIMSK), 23-176
- MMC IPC Rx Interrupt Register, EMAC (EMAC\_IPC\_RXINT), 23-181
- MMC Rx Interrupt Mask Register, EMAC (EMAC\_MMC\_RXIMSK), 23-139
- MMC Rx Interrupt Register, EMAC (EMAC\_MMC\_RXINT), 23-131
- MMC TX Interrupt Mask Register, EMAC (EMAC\_MMC\_TXIMSK), 23-143
- MMC Tx Interrupt Register, EMAC (EMAC\_MMC\_TXINT), 23-135
- MPn Receive Function Address Register, USB (USB\_MPn\_RXFUNCADDR), 22-130
- MPn Receive Hub Address Register, USB (USB\_MPn\_RXHUBADDR), 22-130
- MPn Receive Hub Port Register, USB

(USB\_MPn\_RXHUBPORT), 22-131  
 MPn Transmit Function Address Register, USB  
 (USB\_MPn\_TXFUNCADDR), 22-128  
 MPn Transmit Hub Address Register, USB  
 (USB\_MPn\_TXHUBADDR), 22-128  
 MPn Transmit Hub Port Register, USB  
 (USB\_MPn\_TXHUBPORT), 22-129

## N

NMI (Core 0) Slave 0 (Slaves), 8-7  
 NMI (Core 0) Slave 1 (Slaves), 8-7  
 NMI (Core 1) Slave 0 (Slaves), 8-7  
 NMI (Core 1) Slave 1 (Slaves), 8-7

## O

OPF3 (Memory Pipe) Configuration, PVP  
 (PVP\_OPF3\_CFG), 30-137  
 OPF3 (Memory Pipe) Control, PVP  
 (PVP\_OPF3\_CTL), 30-139  
 OPFn (Camera Pipe) Configuration, PVP  
 (PVP\_OPFn\_CFG), 30-133  
 OPFn (Camera Pipe) Control, PVP  
 (PVP\_OPFn\_CTL), 30-135  
 Outer Loop Address Increment (2D only), DMA  
 (DMA\_YMOD), 13-59  
 Outer Loop Count Start Value (2D only), DMA  
 (DMA\_YCNT), 13-58  
 Overlay A Horizontal End Register, PIXC  
 (PIXC\_HEND\_A), 32-41  
 Overlay A Horizontal Start Register, PIXC  
 (PIXC\_HSTART\_A), 32-40  
 Overlay A Transparency Ratio Register, PIXC  
 (PIXC\_TRANSP\_A), 32-42  
 Overlay A Vertical End Register, PIXC  
 (PIXC\_VEND\_A), 32-42  
 Overlay A Vertical Start Register, PIXC  
 (PIXC\_VSTART\_A), 32-41  
 Overlay B Horizontal End Register, PIXC  
 (PIXC\_HEND\_B), 32-43  
 Overlay B Horizontal Start Register, PIXC  
 (PIXC\_HSTART\_B), 32-43  
 Overlay B Transparency Ratio Register, PIXC  
 (PIXC\_TRANSP\_B), 32-45  
 Overlay B Vertical End Register, PIXC

(PIXC\_VEND\_B), 32-44  
 Overlay B Vertical Start Register, PIXC  
 (PIXC\_VSTART\_B), 32-44  
 Overwrite Protection/Single Shot Transmission 1 Register, CAN (CAN\_OPSS1), 21-39  
 Overwrite Protection/Single Shot Transmission 2 Register, CAN (CAN\_OPSS2), 21-46

## P

PAD Control Register, DMC (DMC\_PADCTL), 11-47  
 PEC Configuration, PVP (PVP\_PEC\_CFG), 30-141  
 PEC Control, PVP (PVP\_PEC\_CTL), 30-142  
 PEC Lower Hysteresis Threshold, PVP  
 (PVP\_PEC\_D1TH0), 30-144  
 PEC Strong Zero Crossing Threshold, PVP  
 (PVP\_PEC\_D2TH1), 30-145  
 PEC Upper Hysteresis Threshold, PVP  
 (PVP\_PEC\_D1TH1), 30-144  
 PEC Weak Zero Crossing Threshold, PVP  
 (PVP\_PEC\_D2TH0), 30-145  
 Peripheral Identification Register, RSI (RSI\_PID0),  
 24-73  
 Peripheral Identification Register, RSI (RSI\_PID1),  
 24-73  
 Peripheral Identification Register, RSI (RSI\_PID2),  
 24-74  
 Peripheral Identification Register, RSI (RSI\_PID3),  
 24-74  
 PHY Control 1 Register, DMC (DMC\_PHY\_CTL1),  
 11-44  
 PHY Control 3 Register, DMC (DMC\_PHY\_CTL3),  
 11-45  
 PHY Control Register, USB (USB\_PHY\_CTL),  
 22-194  
 Pint Assign Register, PINT (PINT\_ASSIGN), 14-98  
 Pint Edge Clear Register, PINT (PINT\_EDGE\_CLR),  
 14-104  
 Pint Edge Set Register, PINT (PINT\_EDGE\_SET),  
 14-100  
 Pint Invert Clear Register, PINT (PINT\_INV\_CLR),  
 14-112  
 Pint Invert Set Register, PINT (PINT\_INV\_SET),  
 14-108  
 Pint Latch Register, PINT (PINT\_LATCH), 14-120  
 Pint Mask Clear Register, PINT (PINT\_MSK\_CLR),

- 14-91
- Pint Mask Set Register, PINT (PINT\_MSK\_SET), 14-87
- Pint Pinstate Register, PINT (PINT\_PINSTATE), 14-116
- Pint Request Register, PINT (PINT\_REQ), 14-95
- PINT0 Pin Interrupt Block (Masters), 8-2, 14-12
- PINT0 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT1 Pin Interrupt Block (Masters), 8-2, 14-12
- PINT1 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT2 Pin Interrupt Block (Masters), 8-3, 14-12
- PINT2 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT3 Pin Interrupt Block (Masters), 8-3, 14-13
- PINT3 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT4 Pin Interrupt Block (Masters), 8-3, 14-13
- PINT4 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT5 Pin Interrupt Block (Masters), 8-3, 14-13
- PINT5 Pin Interrupt Block interrupt, 7-4, 14-12
- PINT\_ASSIGN (Pint Assign Register, PINT), 14-98
- PINT\_EDGE\_CLR (Pint Edge Clear Register, PINT), 14-104
- PINT\_EDGE\_SET (Pint Edge Set Register, PINT), 14-100
- PINT\_INV\_CLR (Pint Invert Clear Register, PINT), 14-112
- PINT\_INV\_SET (Pint Invert Set Register, PINT), 14-108
- PINT\_LATCH (Pint Latch Register, PINT), 14-120
- PINT\_MSK\_CLR (Pint Mask Clear Register, PINT), 14-91
- PINT\_MSK\_SET (Pint Mask Set Register, PINT), 14-87
- PINT\_PINSTATE (Pint Pinstate Register, PINT), 14-116
- PINT\_REQ (Pint Request Register, PINT), 14-95
- PIXC0 Channel 0 DMA (Masters), 8-5, 32-10
- PIXC0 Channel 0 DMA (Slaves), 8-9, 32-10
- PIXC0 Channel 0 DMA interrupt, 7-8, 32-9
- PIXC0 Channel 1 DMA (Masters), 8-5, 32-10
- PIXC0 Channel 1 DMA (Slaves), 8-9, 32-10
- PIXC0 Channel 1 DMA interrupt, 7-8, 32-9
- PIXC0 Channel 2 DMA (Masters), 8-5, 32-10
- PIXC0 Channel 2 DMA (Slaves), 8-9, 32-10
- PIXC0 Channel 2 DMA interrupt, 7-8, 32-9
- PIXC0 Status interrupt, 7-8, 32-9
- PIXC\_CCBIAS (Conversion Bias Register, PIXC), 32-49
- PIXC\_CONBV (BV Conversion Component Register, PIXC), 32-48
- PIXC\_CONGU (GU Conversion Component Register, PIXC), 32-47
- PIXC\_CONRY (RY Conversion Component Register, PIXC), 32-46
- PIXC\_CTL (Control Register, PIXC), 32-37
- PIXC\_HEND\_A (Overlay A Horizontal End Register, PIXC), 32-41
- PIXC\_HEND\_B (Overlay B Horizontal End Register, PIXC), 32-43
- PIXC\_HSTART\_A (Overlay A Horizontal Start Register, PIXC), 32-40
- PIXC\_HSTART\_B (Overlay B Horizontal Start Register, PIXC), 32-43
- PIXC\_IRQSTAT (Interrupt Status Register, PIXC), 32-45
- PIXC\_LPF (Line Per Frame Register, PIXC), 32-40
- PIXC\_PPL (Pixels Per Line Register, PIXC), 32-39
- PIXC\_REVID (Revision Id, PIXC), 32-50
- PIXC\_TC (Transparency Color Register, PIXC), 32-50
- PIXC TRANSP\_A (Overlay A Transparency Ratio Register, PIXC), 32-42
- PIXC TRANSP\_B (Overlay B Transparency Ratio Register, PIXC), 32-45
- PIXC\_VEND\_A (Overlay A Vertical End Register, PIXC), 32-42
- PIXC\_VEND\_B (Overlay B Vertical End Register, PIXC), 32-44
- PIXC\_VSTART\_A (Overlay A Vertical Start Register, PIXC), 32-41
- PIXC\_VSTART\_B (Overlay B Vertical Start Register, PIXC), 32-44
- Pixels Per Line Register, PIXC (PIXC\_PPL), 32-39
- PLL and Oscillator Control Register, USB (USB\_PLL\_OSC), 22-195
- PMA Configuration, PVP (PVP\_PMA\_CFG), 30-222
- Pointer to Next Initial Descriptor, DMA (DMA\_DSCPTR\_NXT), 13-47
- Polynomial Register, CRC (CRC\_POLY), 12-39
- Port x Function Enable Clear Register, PORT (PORT\_FER\_CLR), 14-27

- Port x Function Enable Register, PORT (PORT\_FER), 14-21
- Port x Function Enable Set Register, PORT (PORT\_FER\_SET), 14-24
- Port x GPIO Data Clear Register, PORT (PORT\_DATA\_CLR), 14-38
- Port x GPIO Data Register, PORT (PORT\_DATA), 14-31
- Port x GPIO Data Set Register, PORT (PORT\_DATA\_SET), 14-34
- Port x GPIO Direction Clear Register, PORT (PORT\_DIR\_CLR), 14-51
- Port x GPIO Direction Register, PORT (PORT\_DIR), 14-43
- Port x GPIO Direction Set Register, PORT (PORT\_DIR\_SET), 14-47
- Port x GPIO Input Enable Clear Register, PORT (PORT\_INEN\_CLR), 14-62
- Port x GPIO Input Enable Register, PORT (PORT\_INEN), 14-55
- Port x GPIO Input Enable Set Register, PORT (PORT\_INEN\_SET), 14-58
- Port x GPIO Input Enable Toggle Register, PORT (PORT\_DATA\_TGL), 14-68
- Port x GPIO Lock Register, PORT (PORT\_LOCK), 14-83
- Port x GPIO Polarity Invert Clear Register, PORT (PORT\_POL\_CLR), 14-80
- Port x GPIO Polarity Invert Register, PORT (PORT\_POL), 14-72
- Port x GPIO Polarity Invert Set Register, PORT (PORT\_POL\_SET), 14-76
- Port x GPIO Revision ID, PORT (PORT\_REVID), 14-85
- Port x Multiplexer Control Register, PORT (PORT\_MUX), 14-66
- PORT\_DATA (Port x GPIO Data Register, PORT), 14-31
- PORT\_DATA\_CLR (Port x GPIO Data Clear Register, PORT), 14-38
- PORT\_DATA\_SET (Port x GPIO Data Set Register, PORT), 14-34
- PORT\_DATA\_TGL (Port x GPIO Input Enable Toggle Register, PORT), 14-68
- PORT\_DIR (Port x GPIO Direction Register, PORT), 14-43
- PORT\_DIR\_CLR (Port x GPIO Direction Clear Register, PORT), 14-51
- PORT\_DIR\_SET (Port x GPIO Direction Set Register, PORT), 14-47
- PORT\_FER (Port x Function Enable Register, PORT), 14-21
- PORT\_FER\_CLR (Port x Function Enable Clear Register, PORT), 14-27
- PORT\_FER\_SET (Port x Function Enable Set Register, PORT), 14-24
- PORT\_INEN (Port x GPIO Input Enable Register, PORT), 14-55
- PORT\_INEN\_CLR (Port x GPIO Input Enable Clear Register, PORT), 14-62
- PORT\_INEN\_SET (Port x GPIO Input Enable Set Register, PORT), 14-58
- PORT\_LOCK (Port x GPIO Lock Register, PORT), 14-83
- PORT\_MUX (Port x Multiplexer Control Register, PORT), 14-66
- PORT\_POL (Port x GPIO Polarity Invert Register, PORT), 14-72
- PORT\_POL\_CLR (Port x GPIO Polarity Invert Clear Register, PORT), 14-80
- PORT\_POL\_SET (Port x GPIO Polarity Invert Set Register, PORT), 14-76
- PORT\_REVID (Port x GPIO Revision ID, PORT), 14-85
- Power and Device Control Register, USB (USB\_POWER), 22-89
- Power Good Counter Register, DPM (DPM\_PGCNTR), 5-26
- PPS Control Register, EMAC (EMAC\_TM\_PPSCCTL), 23-217
- PPS Width Register, EMAC (EMAC\_TM\_PPSWIDTH), 23-222
- Previous Initial Descriptor Pointer, DMA (DMA\_DSCPTR\_PRV), 13-61
- Priority ID Mask Register, DMC (DMC\_PRIOMSK), 11-27
- Priority ID Register, DMC (DMC\_PRIO), 11-26
- PVP0 Camera Pipe Control In DMA Channel (Masters), 8-5, 30-9
- PVP0 Camera Pipe Control In DMA Channel (Slaves), 14-43

- 8-10, 30-10
- PVP0 Camera Pipe Control In DMA Channel interrupt, 7-8, 30-8
- PVP0 Camera Pipe Data Out A DMA Channel (Masters), 8-5, 30-9
- PVP0 Camera Pipe Data Out A DMA Channel (Slaves), 8-10, 30-10
- PVP0 Camera Pipe Data Out A DMA Channel interrupt, 7-9, 30-8
- PVP0 Camera Pipe Data Out B DMA Channel (Masters), 8-5, 30-9
- PVP0 Camera Pipe Data Out B DMA Channel (Slaves), 8-9, 30-10
- PVP0 Camera Pipe Data Out B DMA Channel interrupt, 7-8, 30-8
- PVP0 Camera Pipe Data Out C DMA Channel (Masters), 8-5, 30-9
- PVP0 Camera Pipe Data Out C DMA Channel (Slaves), 8-10, 30-10
- PVP0 Camera Pipe Data Out C DMA Channel interrupt, 7-8, 30-8
- PVP0 Camera Pipe Status Out DMA Channel (Masters), 8-5, 30-9
- PVP0 Camera Pipe Status Out DMA Channel (Slaves), 8-10, 30-10
- PVP0 Camera Pipe Status Out DMA Channel interrupt, 7-8, 30-8
- PVP0 Memory Pipe Control In DMA Channel (Masters), 8-5, 30-9
- PVP0 Memory Pipe Control In DMA Channel (Slaves), 8-10, 30-10
- PVP0 Memory Pipe Control In DMA Channel interrupt, 7-9, 30-8
- PVP0 Memory Pipe Data In DMA Channel (Masters), 8-5, 30-9
- PVP0 Memory Pipe Data In DMA Channel (Slaves), 8-10, 30-10
- PVP0 Memory Pipe Data In DMA Channel interrupt, 7-9, 30-8
- PVP0 Memory Pipe Data Out DMA Channel (Masters), 8-5, 30-9
- PVP0 Memory Pipe Data Out DMA Channel (Slaves), 8-10, 30-10
- PVP0 Memory Pipe Data Out DMA Channel interrupt, 7-9, 30-8
- PVP0 Memory Pipe Status Out DMA Channel (Masters), 8-5, 30-9
- PVP0 Memory Pipe Status Out DMA Channel (Slaves), 8-10, 30-10
- PVP0 Memory Pipe Status Out DMA Channel interrupt, 7-9, 30-8
- PVP0 Status 0 (Masters), 8-6, 30-9
- PVP0 Status 0 interrupt, 7-9, 30-8
- PVP0 Status 1 (Masters), 8-6, 30-9
- PVP0 Status 1 interrupt, 7-9, 30-8
- PVP\_ACU\_CFG (ACU Configuration, PVP), 30-152
- PVP\_ACU\_CTL (ACU Control, PVP), 30-154
- PVP\_ACU\_FACTOR (ACU PROD Constant, PVP), 30-158
- PVP\_ACU\_MAX (ACU Upper Sat Threshold Max, PVP), 30-159
- PVP\_ACU\_MIN (ACU Lower Sat Threshold Min, PVP), 30-159
- PVP\_ACU\_OFFSET (ACU SUM Constant, PVP), 30-157
- PVP\_ACU\_SHIFT (ACU Shift Constant, PVP), 30-158
- PVP\_CNVn\_C00C01 (CNVn Coefficients 0,0 and 0,1, PVP), 30-177
- PVP\_CNVn\_C02C03 (CNVn Coefficients 0,2 and 0,3, PVP), 30-178
- PVP\_CNVn\_C04 (CNVn Coefficient 0,4, PVP), 30-179
- PVP\_CNVn\_C10C11 (CNVn Coefficients 1,0 and 1,1, PVP), 30-179
- PVP\_CNVn\_C12C13 (CNVn Coefficients 1,2 and 1,3, PVP), 30-180
- PVP\_CNVn\_C14 (CNVn Coefficient 1,4, PVP), 30-180
- PVP\_CNVn\_C20C21 (CNVn Coefficients 2,0 and 2,1, PVP), 30-181
- PVP\_CNVn\_C22C23 (CNVn Coefficients 2,2 and 2,3, PVP), 30-182
- PVP\_CNVn\_C24 (CNVn Coefficient 2,4, PVP), 30-182
- PVP\_CNVn\_C30C31 (CNVn Coefficients 3,0 and 3,1, PVP), 30-183
- PVP\_CNVn\_C32C33 (CNVn Coefficients 3,2 and 3,3, PVP), 30-183
- PVP\_CNVn\_C34 (CNVn Coefficient 3,4, PVP), 30-183



- 30-184
- PVP\_CNVn\_C40C41 (CNVn Coefficients 4,0 and 4,1, PVP), 30-185
- PVP\_CNVn\_C42C43 (CNVn Coefficients 4,2 and 4,3, PVP), 30-185
- PVP\_CNVn\_C44 (CNVn Coefficient 4,4, PVP), 30-186
- PVP\_CNVn\_CFG (CNVn Configuration, PVP), 30-175
- PVP\_CNVn\_CTL (CNVn Control, PVP), 30-176
- PVP\_CNVn\_SCALE (CNVn Scaling Factor, PVP), 30-187
- PVP\_CTL (Control, PVP), 30-111
- PVP\_IIMn\_CFG (IIMn Configuration, PVP), 30-146
- PVP\_IIMn\_CTL (IIMn Control, PVP), 30-148
- PVP\_IIMn\_SCALE (IIMn Scaling Values, PVP), 30-149
- PVP\_IIMn\_SOVF\_STAT (IIMn Signed Overflow Status, PVP), 30-150
- PVP\_IIMn\_UOVF\_STAT (IIMn Unsigned Overflow Status, PVP), 30-151
- PVP\_ILAT (Interrupt Latch Status n, PVP), 30-125
- PVP\_IMSKn (Interrupt Mask n, PVP), 30-113
- PVP\_IPF0\_CFG (IPF0 (Camera Pipe) Configuration, PVP), 30-164
- PVP\_IPF0\_HPOS (IPF0 (Camera Pipe) Horizontal Position, PVP), 30-172
- PVP\_IPF0\_VPOS (IPF0 (Camera Pipe) Vertical Position, PVP), 30-173
- PVP\_IPF1\_CFG (IPF1 (Memory Pipe) Configuration, PVP), 30-174
- PVP\_IPFn\_CTL (IPFn (Camera/Memory Pipe) Control, PVP), 30-166
- PVP\_IPFn\_FCNT (IPFn (Camera/Memory Pipe) Frame Count, PVP), 30-170
- PVP\_IPFn\_HCNT (IPFn (Camera/Memory Pipe) Horizontal Count, PVP), 30-171
- PVP\_IPFn\_PIPECTL (IPFn (Camera/Memory Pipe) Pipe Control, PVP), 30-165
- PVP\_IPFn\_TAG (IPFn (Camera/Memory Pipe) TAG Value, PVP), 30-170
- PVP\_IPFn\_TAG\_STAT (IPFn (Camera/Memory Pipe) TAG Status, PVP), 30-173
- PVP\_IPFn\_VCNT (IPFn (Camera/Memory Pipe) Vertical Count, PVP), 30-171
- PVP\_IREQn (Interrupt Request n, PVP), 30-130
- PVP\_OPF3\_CFG (OPF3 (Memory Pipe) Configuration, PVP), 30-137
- PVP\_OPF3\_CTL (OPF3 (Memory Pipe) Control, PVP), 30-139
- PVP\_OPFn\_CFG (OPFn (Camera Pipe) Configuration, PVP), 30-133
- PVP\_OPFn\_CTL (OPFn (Camera Pipe) Control, PVP), 30-135
- PVP\_PEC\_CFG (PEC Configuration, PVP), 30-141
- PVP\_PEC\_CTL (PEC Control, PVP), 30-142
- PVP\_PEC\_D1TH0 (PEC Lower Hysteresis Threshold, PVP), 30-144
- PVP\_PEC\_D1TH1 (PEC Upper Hysteresis Threshold, PVP), 30-144
- PVP\_PEC\_D2TH0 (PEC Weak Zero Crossing Threshold, PVP), 30-145
- PVP\_PEC\_D2TH1 (PEC Strong Zero Crossing Threshold, PVP), 30-145
- PVP\_PMA\_CFG (PMA Configuration, PVP), 30-222
- PVP\_REVID (Revision ID, PVP), 30-110
- PVP\_STAT (Status, PVP), 30-117
- PVP\_THCn\_CFG (THCn Configuration, PVP), 30-187
- PVP\_THCn\_CMAXTH (THCn Clip Max Threshold, PVP), 30-195
- PVP\_THCn\_CMAXVAL (THCn Max Clip Value, PVP), 30-195
- PVP\_THCn\_CMINTH (THCn Clip Min Threshold, PVP), 30-194
- PVP\_THCn\_CMINVAL (THCn Min Clip Value, PVP), 30-193
- PVP\_THCn\_CTL (THCn Control, PVP), 30-189
- PVP\_THCn\_HCNT0\_STAT (THCn Histogram Counter Value 0, PVP), 30-211
- PVP\_THCn\_HCNT10\_STAT (THCn Histogram Counter Value 10, PVP), 30-217
- PVP\_THCn\_HCNT11\_STAT (THCn Histogram Counter Value 11, PVP), 30-218
- PVP\_THCn\_HCNT12\_STAT (THCn Histogram Counter Value 12, PVP), 30-218
- PVP\_THCn\_HCNT13\_STAT (THCn Histogram Counter Value 13, PVP), 30-219
- PVP\_THCn\_HCNT14\_STAT (THCn Histogram Counter Value 14, PVP), 30-220

- PVP\_THCn\_HCNT15\_STAT (THCn Histogram Counter Value 15, PVP), 30-220
- PVP\_THCn\_HCNT1\_STAT (THCn Histogram Counter Value 1, PVP), 30-212
- PVP\_THCn\_HCNT2\_STAT (THCn Histogram Counter Value 2, PVP), 30-212
- PVP\_THCn\_HCNT3\_STAT (THCn Histogram Counter Value 3, PVP), 30-213
- PVP\_THCn\_HCNT4\_STAT (THCn Histogram Counter Value 4, PVP), 30-214
- PVP\_THCn\_HCNT5\_STAT (THCn Histogram Counter Value 5, PVP), 30-214
- PVP\_THCn\_HCNT6\_STAT (THCn Histogram Counter Value 6, PVP), 30-215
- PVP\_THCn\_HCNT7\_STAT (THCn Histogram Counter Value 7, PVP), 30-215
- PVP\_THCn\_HCNT8\_STAT (THCn Histogram Counter Value 8, PVP), 30-216
- PVP\_THCn\_HCNT9\_STAT (THCn Histogram Counter Value 9, PVP), 30-217
- PVP\_THCn\_HFCNT (THCn Histogram Frame Count, PVP), 30-192
- PVP\_THCn\_HFCNT\_STAT (THCn Histogram Frame Count Status, PVP), 30-211
- PVP\_THCn\_HHCNT (THCn Histogram Horizontal Count, PVP), 30-207
- PVP\_THCn\_HHPOS (THCn Histogram Horizontal Position, PVP), 30-205
- PVP\_THCn\_HVCNT (THCn Histogram Vertical Count, PVP), 30-207
- PVP\_THCn\_HVPOS (THCn Histogram Vertical Position, PVP), 30-206
- PVP\_THCn\_RHCNT (THCn RLE Horizontal Count, PVP), 30-209
- PVP\_THCn\_RHPOS (THCn RLE Horizontal Position, PVP), 30-208
- PVP\_THCn\_RMAXREP (THCn Max RLE Reports, PVP), 30-193
- PVP\_THCn\_RREP\_STAT (THCn Number of RLE Reports, PVP), 30-221
- PVP\_THCn\_RVCNT (THCn RLE Vertical Count, PVP), 30-210
- PVP\_THCn\_RVPOS (THCn RLE Vertical Position, PVP), 30-209
- PVP\_THCn\_TH0 (THCn Threshold Value 0, PVP), 30-196
- PVP\_THCn\_TH1 (THCn Threshold Value 1, PVP), 30-196
- PVP\_THCn\_TH10 (THCn Threshold Value 10, PVP), 30-202
- PVP\_THCn\_TH11 (THCn Threshold Value 11, PVP), 30-202
- PVP\_THCn\_TH12 (THCn Threshold Value 12, PVP), 30-203
- PVP\_THCn\_TH13 (THCn Threshold Value 13, PVP), 30-204
- PVP\_THCn\_TH14 (THCn Threshold Value 14, PVP), 30-204
- PVP\_THCn\_TH15 (THCn Threshold Value 15, PVP), 30-205
- PVP\_THCn\_TH2 (THCn Threshold Value 2, PVP), 30-197
- PVP\_THCn\_TH3 (THCn Threshold Value 3, PVP), 30-198
- PVP\_THCn\_TH4 (THCn Threshold Value 4, PVP), 30-198
- PVP\_THCn\_TH5 (THCn Threshold Value 5, PVP), 30-199
- PVP\_THCn\_TH6 (THCn Threshold Value 6, PVP), 30-199
- PVP\_THCn\_TH7 (THCn Threshold Value 7, PVP), 30-200
- PVP\_THCn\_TH8 (THCn Threshold Value 8, PVP), 30-201
- PVP\_THCn\_TH9 (THCn Threshold Value 9, PVP), 30-201
- PVP\_UDS\_CFG (UDS Configuration, PVP), 30-160
- PVP\_UDS\_CTL (UDS Control, PVP), 30-161
- PVP\_UDS\_HAVG (UDS HAVG, PVP), 30-163
- PVP\_UDS\_OHCNT (UDS Output HCNT, PVP), 30-162
- PVP\_UDS\_OVCNT (UDS Output VCNT, PVP), 30-162
- PVP\_UDS\_VAVG (UDS VAVG, PVP), 30-164
- PWM0 PWMTMR Group (Masters), 8-3, 18-4
- PWM0 PWMTMR Group interrupt, 7-4, 18-3
- PWM0 Trip interrupt, 7-4, 18-3
- PWM1 PWMTMR Group (Masters), 8-3, 18-4
- PWM1 PWMTMR Group interrupt, 7-4, 18-3
- PWM1 Trip interrupt, 7-4, 18-3

- PWM\_ACTL (Channel A Control Register, PWM), 18-76
- PWM\_AH0 (Channel A-High Duty-0 Register, PWM), 18-80
- PWM\_AH1 (Channel A-High Duty-1 Register, PWM), 18-80
- PWM\_AL0 (Channel A-Low Duty-0 Register, PWM), 18-82
- PWM\_AL1 (Channel A-Low Duty-1 Register, PWM), 18-82
- PWM\_BCTL (Channel B Control Register, PWM), 18-83
- PWM\_BH0 (Channel B-High Duty-0 Register, PWM), 18-87
- PWM\_BH1 (Channel B-High Duty-1 Register, PWM), 18-87
- PWM\_BL0 (Channel B-Low Duty-0 Register, PWM), 18-88
- PWM\_BL1 (Channel B-Low Duty-1 Register, PWM), 18-89
- PWM\_CCTL (Channel C Control Register, PWM), 18-90
- PWM\_CH0 (Channel C-High Pulse Duty Register 0, PWM), 18-93
- PWM\_CH1 (Channel C-High Pulse Duty Register 1, PWM), 18-93
- PWM\_CHANCFG (Channel Config Register, PWM), 18-44
- PWM\_CHOPCFG (Chop Configuration Register, PWM), 18-66
- PWM\_CL0 (Channel C-Low Pulse Duty Register 0, PWM), 18-95
- PWM\_CL1 (Channel C-Low Duty-1 Register, PWM), 18-95
- PWM\_CTL (Control Register, PWM), 18-40
- PWM\_DCTL (Channel D Control Register, PWM), 18-96
- PWM\_DH0 (Channel D-High Duty-0 Register, PWM), 18-99
- PWM\_DH1 (Channel D-High Pulse Duty Register 1, PWM), 18-99
- PWM\_DL0 (Channel D-Low Pulse Duty Register 0, PWM), 18-101
- PWM\_DL1 (Channel D-Low Pulse Duty Register 1, PWM), 18-101
- PWM\_DLYA (Channel A Delay Register, PWM), 18-73
- PWM\_DLYB (Channel B Delay Register, PWM), 18-74
- PWM\_DLYC (Channel C Delay Register, PWM), 18-75
- PWM\_DLYD (Channel D Delay Register, PWM), 18-76
- PWM\_DT (Dead Time Register, PWM), 18-67
- PWM\_ILAT (Interrupt Latch Register, PWM), 18-64
- PWM\_IMSK (Interrupt Mask Register, PWM), 18-62
- PWM\_STAT (Status Register, PWM), 18-56
- PWM\_SYNC\_WID (Sync Pulse Width Register, PWM), 18-68
- PWM\_TM0 (Timer 0 Period Register, PWM), 18-69
- PWM\_TM1 (Timer 1 Period Register, PWM), 18-70
- PWM\_TM2 (Timer 2 Period Register, PWM), 18-71
- PWM\_TM3 (Timer 3 Period Register, PWM), 18-72
- PWM\_TM4 (Timer 4 Period Register, PWM), 18-73
- PWM\_TRIPCFG (Trip Config Register, PWM), 18-51
- ## R
- RAM Information Register, USB (USB\_RAMINFO), 22-123
- RCU0 System Reset 0 (Slaves), 8-7, 33-2
- RCU0 System Reset 1 (Slaves), 8-7, 33-2
- RCU\_BCODE (Boot Code Register, RCU), 33-14, 34-72
- RCU\_CRCTL (Core Reset Control Register, RCU), 33-10
- RCU\_CRSTAT (Core Reset Status Register, RCU), 33-11
- RCU\_CTL (Control Register, RCU), 33-6
- RCU\_SIDIS (System Interface Disable Register, RCU), 33-11
- RCU\_SISTAT (System Interface Status Register, RCU), 33-12
- RCU\_STAT (Status Register, RCU), 33-7, 34-69
- RCU\_SVECT\_LCK (SVECT Lock Register, RCU), 33-13
- RCU\_SVECTn (Software Vector Register n, RCU), 33-15, 34-71
- Read Priority Count Register, L2CTL (L2CTL\_RPCR), 10-23
- Read Wait Enable Register, RSI (RSI\_RD\_WAIT),

- 24-72
- Receive Buffer Register, UART (UART\_RBR), 19-49
- Receive Buffer, LP (LP\_RX), 28-24
- Receive Control Register, SPI (SPI\_RXCTL), 25-31
- Receive Counter Register, UART (UART\_RXCNT), 19-53
- Receive FIFO Address Register, USB (USB\_RXFIFOADDR), 22-121
- Receive FIFO Data Register, SPI (SPI\_RFIFO), 25-60
- Receive FIFO Size Register, USB (USB\_RXFIFOSZ), 22-119
- Receive Interrupt Enable Register, USB (USB\_INTRRXE), 22-102
- Receive Interrupt Register, USB (USB\_INTRRX), 22-96
- Receive Message Lost 1 Register, CAN (CAN\_RML1), 21-36
- Receive Message Lost 2 Register, CAN (CAN\_RML2), 21-43
- Receive Message Pending 1 Register, CAN (CAN\_RMP1), 21-35
- Receive Message Pending 2 Register, CAN (CAN\_RMP2), 21-42
- Receive Shift Register, UART (UART\_RSR), 19-52
- Received Word Count Register, SPI (SPI\_RWC), 25-44
- Received Word Count Reload Register, SPI (SPI\_RWCR), 25-44
- Refresh Address Register, L2CTL (L2CTL\_RFA), 10-25
- registers
  - diagram conventions, -lxxxvii
- Remote Frame Handling 1 Register, CAN (CAN\_RFH1), 21-38
- Remote Frame Handling 2 Register, CAN (CAN\_RFH2), 21-45
- Reserved (Masters), 8-2, 8-3, 8-6
- Reserved (Slaves), 8-7, 8-8, 8-10, 8-11
- Reserved interrupt, 7-6, 7-9
- Response 0 Register, RSI (RSI\_RESP0), 24-49
- Response 1 Register, RSI (RSI\_RESP1), 24-49
- Response 2 Register, RSI (RSI\_RESP2), 24-50
- Response 3 Register, RSI (RSI\_RESP3), 24-50
- Response Command Register, RSI (RSI\_RESP\_CMD), 24-48
- Restore n Register, DPM (DPM\_RESTOREn), 5-28
- Revision ID Register, CRC (CRC\_REVID), 12-44
- Revision ID Register, TIMER (TIMER\_REVID), 15-29
- Revision ID Register, TRU (TRU\_REVID), 8-18
- Revision ID Register, UART (UART\_REVID), 19-25
- Revision Id, PIXC (PIXC\_REVID), 32-50
- Revision ID, PVP (PVP\_REVID), 30-110
- RSI0 DMA Channel (Masters), 8-3, 24-4
- RSI0 DMA Channel (Slaves), 8-8, 24-5
- RSI0 DMA Channel interrupt, 7-6, 24-4
- RSI0 Interrupt 0 interrupt, 7-6, 24-4
- RSI0 Interrupt 1 interrupt, 7-6, 24-4
- RSI\_ARG (Argument Register, RSI), 24-46
- RSI\_BACK\_TOUT (Boot Acknowledge Timeout Register, RSI), 24-62
- RSI\_BLKSZ (Block Size Register, RSI), 24-63
- RSI\_BOOT\_TCNTR (Boot Timing Counter Register, RSI), 24-62
- RSI\_CEATA (This register contains bit to dis CCS gen, RSI), 24-61
- RSI\_CFG (Configuration Register, RSI), 24-70
- RSI\_CMD (Command Register, RSI), 24-46
- RSI\_CTL (Control Register, RSI), 24-44
- RSI\_DATA\_CNT (Data Count Register, RSI), 24-53
- RSI\_DATA\_CTL (Data Control Register, RSI), 24-52
- RSI\_DATA\_LEN (Data Length Register, RSI), 24-52
- RSI\_DATA\_TMR (Data Timer Register, RSI), 24-51
- RSI\_FIFO (Data FIFO Register, RSI), 24-64
- RSI\_FIFO\_CNT (FIFO Counter Register, RSI), 24-61
- RSI\_IMSK0 (Exception Mask Register, RSI), 24-67
- RSI\_PID0 (Peripheral Identification Register, RSI), 24-73
- RSI\_PID1 (Peripheral Identification Register, RSI), 24-73
- RSI\_PID2 (Peripheral Identification Register, RSI), 24-74
- RSI\_PID3 (Peripheral Identification Register, RSI), 24-74
- RSI\_RD\_WAIT (Read Wait Enable Register, RSI), 24-72
- RSI\_RESP0 (Response 0 Register, RSI), 24-49
- RSI\_RESP1 (Response 1 Register, RSI), 24-49
- RSI\_RESP2 (Response 2 Register, RSI), 24-50
- RSI\_RESP3 (Response 3 Register, RSI), 24-50

- RSI\_RESP\_CMD (Response Command Register, RSI), 24-48
- RSI\_SLP\_WKUP\_TOUT (Sleep Wakeup Timeout Register, RSI), 24-63
- RSI\_STAT0 (Exception Status Register, RSI), 24-65
- RSI\_XFR\_IMSK0 (Interrupt 0 Mask Register, RSI), 24-57
- RSI\_XFR\_IMSK1 (Interrupt 1 Mask Register, RSI), 24-59
- RSI\_XFRSTAT (Status Register, RSI), 24-54
- RSI\_XFRSTAT\_CLR (Status Clear Register, RSI), 24-56
- Run Clear Register, TIMER (TIMER\_RUN\_CLR), 15-31
- Run Register, TIMER (TIMER\_RUN), 15-30
- Run Set Register, TIMER (TIMER\_RUN\_SET), 15-30
- Rx 1024- to Max-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX1024TOMAX\_GB), 23-171
- Rx 128- to 255-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX128TO255\_GB), 23-169
- Rx 256- to 511-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX256TO511\_GB), 23-170
- Rx 512- to 1023-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX512TO1023\_GB), 23-170
- Rx 64-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX64\_GB), 23-168
- Rx 65- to 127-Byte Frames (Good/Bad) Register, EMAC (EMAC\_RX65TO127\_GB), 23-168
- Rx alignment Error Register, EMAC (EMAC\_RXALIGN\_ERR), 23-165
- Rx Broadcast Frames (Good) Register, EMAC (EMAC\_RXBCASTFRM\_G), 23-163
- Rx CRC Error Register, EMAC (EMAC\_RXCRC\_ERR), 23-164
- Rx Data Double-Byte Register, TWI (TWI\_RXDATA16), 20-42
- Rx Data Single-Byte Register, TWI (TWI\_RXDATA8), 20-41
- Rx FIFO Overflow Register, EMAC (EMAC\_RXFIFO\_OVF), 23-174
- Rx Frame Count (Good/Bad) Register, EMAC (EMAC\_RXFRMCNT\_GB), 23-161
- Rx ICMP Error Frames Register, EMAC (EMAC\_RXICMP\_ERR\_FRM), 23-193
- Rx ICMP Error Octets Register, EMAC (EMAC\_RXICMP\_ERR\_OCT), 23-202
- Rx ICMP Good Frames Register, EMAC (EMAC\_RXICMP\_GD\_FRM), 23-193
- Rx ICMP Good Octets Register, EMAC (EMAC\_RXICMP\_GD\_OCT), 23-201
- Rx IPv4 Datagrams (Good) Register, EMAC (EMAC\_RXIPV4\_GD\_FRM), 23-185
- Rx IPv4 Datagrams Fragmented Frames Register, EMAC (EMAC\_RXIPV4\_FRAG\_FRM), 23-187
- Rx IPv4 Datagrams Fragmented Octets Register, EMAC (EMAC\_RXIPV4\_FRAG\_OCT), 23-196
- Rx IPv4 Datagrams Good Octets Register, EMAC (EMAC\_RXIPV4\_GD\_OCT), 23-194
- Rx IPv4 Datagrams Header Errors Register, EMAC (EMAC\_RXIPV4\_HDR\_ERR\_FRM), 23-186
- Rx IPv4 Datagrams Header Errors Register, EMAC (EMAC\_RXIPV4\_HDR\_ERR\_OCT), 23-194
- Rx IPv4 Datagrams No Payload Frame Register, EMAC (EMAC\_RXIPV4\_NOPAY\_FRM), 23-187
- Rx IPv4 Datagrams No Payload Octets Register, EMAC (EMAC\_RXIPV4\_NOPAY\_OCT), 23-195
- Rx IPv4 UDP Disabled Frames Register, EMAC (EMAC\_RXIPV4\_UDSBL\_FRM), 23-188
- Rx IPv4 UDP Disabled Octets Register, EMAC (EMAC\_RXIPV4\_UDSBL\_OCT), 23-196
- Rx IPv6 Datagrams Good Frames Register, EMAC (EMAC\_RXIPV6\_GD\_FRM), 23-188
- Rx IPv6 Datagrams Header Error Frames Register, EMAC (EMAC\_RXIPV6\_HDR\_ERR\_FRM), 23-189
- Rx IPv6 Datagrams No Payload Frames Register, EMAC (EMAC\_RXIPV6\_NOPAY\_FRM), 23-190
- Rx IPv6 Good Octets Register, EMAC (EMAC\_RXIPV6\_GD\_OCT), 23-197
- Rx IPv6 Header Errors Register, EMAC (EMAC\_RXIPV6\_HDR\_ERR\_OCT), 23-197
- Rx IPv6 No Payload Octets Register, EMAC (EMAC\_RXIPV6\_NOPAY\_OCT), 23-198
- Rx Jab Error Register, EMAC (EMAC\_RXJAB\_ERR), 23-166
- Rx Length Error Register, EMAC (EMAC\_RXLEN\_ERR), 23-172
- Rx Multicast Frames (Good) Register, EMAC (EMAC\_RXMCASTFRM\_G), 23-164
- Rx Octet Count (Good) Register, EMAC

(EMAC\_RXOCTCNT\_G), 23-162  
 Rx Octet Count (Good/Bad) Register, EMAC (EMAC\_RXOCTCNT\_GB), 23-162  
 Rx Out Of Range Type Register, EMAC (EMAC\_RXOORTYPE), 23-173  
 Rx Oversize (Good) Register, EMAC (EMAC\_RXOSIZE\_G), 23-167  
 Rx Pause Frames Register, EMAC (EMAC\_RXPAUSEFRM), 23-173  
 Rx Runt Error Register, EMAC (EMAC\_RXRUNT\_ERR), 23-165  
 Rx TCP Error Frames Register, EMAC (EMAC\_RXTCP\_ERR\_FRM), 23-192  
 Rx TCP Error Octets Register, EMAC (EMAC\_RXTCP\_ERR\_OCT), 23-200  
 Rx TCP Good Frames Register, EMAC (EMAC\_RXTCP\_GD\_FRM), 23-191  
 Rx TCP Good Octets Register, EMAC (EMAC\_RXTCP\_GD\_OCT), 23-200  
 Rx UDP Error Frames Register, EMAC (EMAC\_RXUDP\_ERR\_FRM), 23-191  
 Rx UDP Error Octets Register, EMAC (EMAC\_RXUDP\_ERR\_OCT), 23-199  
 Rx UDP Good Frames Register, EMAC (EMAC\_RXUDP\_GD\_FRM), 23-190  
 Rx UDP Good Octets Register, EMAC (EMAC\_RXUDP\_GD\_OCT), 23-199  
 Rx Undersize (Good) Register, EMAC (EMAC\_RXUSIZE\_G), 23-167  
 Rx Unicast Frames (Good) Register, EMAC (EMAC\_RXUCASTFRM\_G), 23-171  
 Rx VLAN Frames (Good/Bad) Register, EMAC (EMAC\_RXVLANFRM\_GB), 23-174  
 Rx Watch Dog Error Register, EMAC (EMAC\_RXWDOG\_ERR), 23-175  
 RY Conversion Component Register, PIXC (PIXC\_CONRY), 32-46

## S

Samples Per Line Register, EPPI (EPPI\_LINE), 31-75  
 SCB\_ARBRn (Arbitration Read Channel Master Interface n Register, SCB), 2-22  
 SCB\_ARBWn (Arbitration Write Channel Master Interface n Register, SCB), 2-23  
 SCB\_MASTERS (Master Interfaces Number Register,

SCB), 2-24  
 SCB\_SLAVES (Slave Interfaces Number Register, SCB), 2-23  
 SCI Active Register n, SEC (SEC\_CACTn), 7-23  
 SCI Control Register n, SEC (SEC\_CCTLn), 7-19  
 SCI Group Mask Register n, SEC (SEC\_CGMSKn), 7-25  
 SCI Priority Level Register n, SEC (SEC\_CPLVLn), 7-26  
 SCI Priority Mask Register n, SEC (SEC\_CPMSKn), 7-24  
 SCI Source ID Register n, SEC (SEC\_CSIDn), 7-28  
 SCI Status Register n, SEC (SEC\_CSTATn), 7-20  
 SCL Clock Divider Register, TWI (TWI\_CLKDIV), 20-19  
 Scratch Register, UART (UART\_SCR), 19-39  
 SDU0 DMA (Masters), 8-3, 35-3  
 SDU0 DMA (Slaves), 8-8, 35-3  
 SDU0 DMA interrupt, 7-6, 35-2  
 SDU0 Slave Trigger (Slaves), 8-10, 35-3  
 SDU\_CTL (Control Register, SDU), 35-14  
 SDU\_DMARD (DMA Read Data Register, SDU), 35-24  
 SDU\_DMAWD (DMA Write Data Register, SDU), 35-24  
 SDU\_GHLT (Group Halt Register, SDU), 35-28  
 SDU\_IDCODE (ID Code Register, SDU), 35-13  
 SDU\_MACADDR (Memory Access Address Register, SDU), 35-23  
 SDU\_MACCTL (Memory Access Control Register, SDU), 35-21  
 SDU\_MACDATA (Memory Access Data Register, SDU), 35-23  
 SDU\_MSG (Message Register, SDU), 35-25  
 SDU\_MSG\_CLR (Message Clear Register, SDU), 35-27  
 SDU\_MSG\_SET (Message Set Register, SDU), 35-26  
 SDU\_STAT (Status Register, SDU), 35-16  
 SEC0 Error interrupt, 7-3  
 SEC0 Fault (Masters), 8-6  
 SEC\_CACTn (SCI Active Register n, SEC), 7-23  
 SEC\_CCTLn (SCI Control Register n, SEC), 7-19  
 SEC\_CGMSKn (SCI Group Mask Register n, SEC), 7-25  
 SEC\_CPLVLn (SCI Priority Level Register n, SEC),

- 7-26
- SEC\_CPMSKn (SCI Priority Mask Register n, SEC), 7-24
- SEC\_CPNDn (Core Pending Register n, SEC), 7-23
- SEC\_CSIDn (SCI Source ID Register n, SEC), 7-28
- SEC\_CSTATn (SCI Status Register n, SEC), 7-20
- SEC\_END (Global End Register, SEC), 7-43
- SEC\_FCOPP (Fault COP Period Register, SEC), 7-38
- SEC\_FCOPP\_CUR (Fault COP Period Current Register, SEC), 7-39
- SEC\_FCTL (Fault Control Register, SEC), 7-28
- SEC\_FDLY (Fault Delay Register, SEC), 7-36
- SEC\_FDLY\_CUR (Fault Delay Current Register, SEC), 7-36
- SEC\_FEND (Fault End Register, SEC), 7-35
- SEC\_FSID (Fault Source ID Register, SEC), 7-34
- SEC\_FSRDLY (Fault System Reset Delay Register, SEC), 7-37
- SEC\_FSRDLY\_CUR (Fault System Reset Delay Current Register, SEC), 7-38
- SEC\_FSTAT (Fault Status Register, SEC), 7-31
- SEC\_GCTL (Global Control Register, SEC), 7-40
- SEC\_GSTAT (Global Status Register, SEC), 7-41
- SEC\_RAISE (Global Raise Register, SEC), 7-43
- SEC\_SCTLn (Source Control Register n, SEC), 7-44
- SEC\_SSTATn (Source Status Register n, SEC), 7-47
- Shadow EMR1 Register, DMC (DMC\_EMR1), 11-38
- Shadow EMR2 Register, DMC (DMC\_EMR2), 11-40
- Shadow EMR3 Register, DMC (DMC\_EMR3), 11-42
- Shadow Input Transmit Buffer, LP (LP\_TXIN\_SHDW), 28-25
- Shadow MR Register, DMC (DMC\_MR), 11-35
- Shadow Output Transmit Buffer, LP (LP\_TXOUT\_SHDW), 28-26
- Slave Interfaces Number Register, SCB (SCB\_SLAVES), 2-23
- Slave Mode Address Register, TWI (TWI\_SLVADDR), 20-23
- Slave Mode Control Register, TWI (TWI\_SLVCTL), 20-21
- Slave Mode Status Register, TWI (TWI\_SLVSTAT), 20-22
- Slave Select Register, SPI (SPI\_SLVSEL), 25-40
- Slave Select Register, TRU (TRU\_SSRn), 8-14
- Sleep Wakeup Timeout Register, RSI (RSI\_SLP\_WKUP\_TOUT), 24-63
- SMC\_B0CTL (Bank 0 Control Register, SMC), 9-26
- SMC\_B0ETIM (Bank 0 Extended Timing Register, SMC), 9-32
- SMC\_B0TIM (Bank 0 Timing Register, SMC), 9-30
- SMC\_B1CTL (Bank 1 Control Register, SMC), 9-34
- SMC\_B1ETIM (Bank 1 Extended Timing Register, SMC), 9-40
- SMC\_B1TIM (Bank 1 Timing Register, SMC), 9-38
- SMC\_B2CTL (Bank 2 Control Register, SMC), 9-42
- SMC\_B2ETIM (Bank 2 Extended Timing Register, SMC), 9-48
- SMC\_B2TIM (Bank 2 Timing Register, SMC), 9-46
- SMC\_B3CTL (Bank 3 Control Register, SMC), 9-50
- SMC\_B3ETIM (Bank 3 Extended Timing Register, SMC), 9-56
- SMC\_B3TIM (Bank 3 Timing Register, SMC), 9-54
- SMC\_GCTL (Grant Control Register, SMC), 9-24
- SMC\_GSTAT (Grant Status Register, SMC), 9-25
- SMI Address Register, EMAC (EMAC\_SMI\_ADDR), 23-117
- SMI Data Register, EMAC (EMAC\_SMI\_DATA), 23-119
- Software Reset Register, USB (USB\_SOFT\_RST), 22-127
- Software Vector Register n, RCU (RCU\_SVECTn), 33-15, 34-71
- Software-driven Interrupt 0 interrupt, 7-4
- Software-driven Interrupt 1 interrupt, 7-4
- Software-driven Interrupt 2 interrupt, 7-4
- Software-driven Interrupt 3 interrupt, 7-5
- Software-driven Trigger 0 (Masters), 8-6
- Software-driven Trigger 1 (Masters), 8-6
- Software-driven Trigger 2 (Masters), 8-6
- Software-driven Trigger 3 (Masters), 8-6
- Software-driven Trigger 4 (Masters), 8-6
- Software-driven Trigger 5 (Masters), 8-6
- Source Control Register n, SEC (SEC\_SCTLn), 7-44
- Source Status Register n, SEC (SEC\_SSTATn), 7-47
- SPI0 RX DMA Channel (Masters), 8-3, 25-4
- SPI0 RX DMA Channel (Slaves), 8-8, 25-4
- SPI0 RX DMA Channel interrupt, 7-5, 25-3
- SPI0 Status interrupt, 7-5, 25-3
- SPI0 TX DMA Channel (Masters), 8-3, 25-4
- SPI0 TX DMA Channel (Slaves), 8-8, 25-4

- SPI0 TX DMA Channel interrupt, 7-5, 25-3
- SPI1 RX DMA Channel (Masters), 8-3, 25-4
- SPI1 RX DMA Channel (Slaves), 8-8, 25-4
- SPI1 RX DMA Channel interrupt, 7-6, 25-4
- SPI1 Status interrupt, 7-6, 25-4
- SPI1 TX DMA Channel (Masters), 8-3, 25-4
- SPI1 TX DMA Channel (Slaves), 8-8, 25-4
- SPI1 TX DMA Channel interrupt, 7-5, 25-3
- SPI\_CLK (Clock Rate Register, SPI), 25-37
- SPI\_CTL (Control Register, SPI), 25-25
- SPI\_DLY (Delay Register, SPI), 25-38
- SPI\_ILAT (Masked Interrupt Condition Register, SPI), 25-57
- SPI\_ILAT\_CLR (Masked Interrupt Clear Register, SPI), 25-59
- SPI\_IMSK (Interrupt Mask Register, SPI), 25-47
- SPI\_IMSK\_CLR (Interrupt Mask Clear Register, SPI), 25-49
- SPI\_IMSK\_SET (Interrupt Mask Set Register, SPI), 25-50
- SPI\_RFIFO (Receive FIFO Data Register, SPI), 25-60
- SPI\_RWC (Received Word Count Register, SPI), 25-44
- SPI\_RWCR (Received Word Count Reload Register, SPI), 25-44
- SPI\_RXCTL (Receive Control Register, SPI), 25-31
- SPI\_SLVSEL (Slave Select Register, SPI), 25-40
- SPI\_STAT (Status Register, SPI), 25-52
- SPI\_TFIFO (Transmit FIFO Data Register, SPI), 25-61
- SPI\_TWC (Transmitted Word Count Register, SPI), 25-45
- SPI\_TWCR (Transmitted Word Count Reload Register, SPI), 25-46
- SPI\_TXCTL (Transmit Control Register, SPI), 25-34
- SPORT 1 Channel B DMA (Masters), 26-10
- SPORT\_CTL\_A (Half SPORT 'A' Control Register, SPORT), 26-52
- SPORT0 Channel A DMA (Masters), 8-3, 26-10
- SPORT0 Channel A DMA (Slaves), 8-7, 26-10
- SPORT0 Channel A DMA interrupt, 7-5, 26-9
- SPORT0 Channel A Status interrupt, 7-5, 26-9
- SPORT0 Channel B DMA (Masters), 8-3, 26-10
- SPORT0 Channel B DMA (Slaves), 8-7, 26-10
- SPORT0 Channel B DMA interrupt, 7-5, 26-9
- SPORT0 Channel B Status interrupt, 7-5, 26-9
- SPORT1 Channel A DMA (Masters), 8-3, 26-10
- SPORT1 Channel A DMA (Slaves), 8-8, 26-10
- SPORT1 Channel A DMA interrupt, 7-5, 26-9
- SPORT1 Channel A Status interrupt, 7-5, 26-9
- SPORT1 Channel B DMA (Masters), 8-3
- SPORT1 Channel B DMA (Slaves), 8-8, 26-10
- SPORT1 Channel B DMA interrupt, 7-5, 26-9
- SPORT1 Channel B Status interrupt, 7-5, 26-9
- SPORT2 Channel A DMA (Masters), 8-3, 26-10
- SPORT2 Channel A DMA (Slaves), 8-8, 26-11
- SPORT2 Channel A DMA interrupt, 7-5, 26-10
- SPORT2 Channel A Status interrupt, 7-5, 26-10
- SPORT2 Channel B DMA (Masters), 8-3, 26-10
- SPORT2 Channel B DMA (Slaves), 8-8, 26-11
- SPORT2 Channel B DMA interrupt, 7-5, 26-10
- SPORT2 Channel B Status interrupt, 7-5, 26-10
- SPORT\_CS0\_A (Half SPORT 'A' Multi-channel 0-31 Select Register, SPORT), 26-65
- SPORT\_CS0\_B (Half SPORT 'B' Multi-channel 0-31 Select Register, SPORT), 26-90
- SPORT\_CS1\_A (Half SPORT 'A' Multi-channel 32-63 Select Register, SPORT), 26-66
- SPORT\_CS1\_B (Half SPORT 'B' Multi-channel 32-63 Select Register, SPORT), 26-91
- SPORT\_CS2\_A (Half SPORT 'A' Multi-channel 64-95 Select Register, SPORT), 26-67
- SPORT\_CS2\_B (Half SPORT 'B' Multichannel 64-95 Select Register, SPORT), 26-92
- SPORT\_CS3\_A (Half SPORT 'A' Multi-channel 96-127 Select Register, SPORT), 26-67
- SPORT\_CS3\_B (Half SPORT 'B' Multichannel 96-127 Select Register, SPORT), 26-92
- SPORT\_CTL2\_A (Half SPORT 'A' Control 2 Register, SPORT), 26-71
- SPORT\_CTL2\_B (Half SPORT 'B' Control 2 Register, SPORT), 26-96
- SPORT\_CTL\_B (Half SPORT 'B' Control Register, SPORT), 26-77
- SPORT\_DIV\_A (Half SPORT 'A' Divisor Register, SPORT), 26-62
- SPORT\_DIV\_B (Half SPORT 'B' Divisor Register, SPORT), 26-87
- SPORT\_ERR\_A (Half SPORT 'A' Error Register, SPORT), 26-68
- SPORT\_ERR\_B (Half SPORT 'B' Error Register,



- SPORT), 26-93
- SPORT\_MCTL\_A (Half SPORT 'A' Multi-channel Control Register, SPORT), 26-63
- SPORT\_MCTL\_B (Half SPORT 'B' Multi-channel Control Register, SPORT), 26-88
- SPORT\_MSTAT\_A (Half SPORT 'A' Multi-channel Status Register, SPORT), 26-71
- SPORT\_MSTAT\_B (Half SPORT 'B' Multi-channel Status Register, SPORT), 26-96
- SPORT\_RXPRI\_A (Half SPORT 'A' Rx Buffer (Primary) Register, SPORT), 26-74
- SPORT\_RXPRI\_B (Half SPORT 'B' Rx Buffer (Primary) Register, SPORT), 26-99
- SPORT\_RXSEC\_A (Half SPORT 'A' Rx Buffer (Secondary) Register, SPORT), 26-76
- SPORT\_RXSEC\_B (Half SPORT 'B' Rx Buffer (Secondary) Register, SPORT), 26-101
- SPORT\_TXPRI\_A (Half SPORT 'A' Tx Buffer (Primary) Register, SPORT), 26-73
- SPORT\_TXPRI\_B (Half SPORT 'B' Tx Buffer (Primary) Register, SPORT), 26-98
- SPORT\_TXSEC\_A (Half SPORT 'A' Tx Buffer (Secondary) Register, SPORT), 26-75
- SPORT\_TXSEC\_B (Half SPORT 'B' Tx Buffer (Secondary) Register, SPORT), 26-100
- SPU\_CTL (Control Register, SPU), 4-8
- SPU\_STAT (Status Register, SPU), 4-9
- SPU\_WPn (Write Protect Register n, SPU), 4-10
- Start Address of Current Buffer, DMA (DMA\_ADDRSTART), 13-47
- Status Clear Register, RSI (RSI\_XFRSTAT\_CLR), 24-56
- Status Information Register, TRU (TRU\_STAT), 8-17
- Status Interrupt Latch Register, TIMER (TIMER\_STAT\_ILAT), 15-37
- Status Interrupt Mask Register, TIMER (TIMER\_STAT\_IMSK), 15-34
- Status Register, CAN (CAN\_STAT), 21-51
- Status Register, CGU (CGU\_STAT), 3-13
- Status Register, CNT (CNT\_STAT), 17-20
- Status Register, CRC (CRC\_STAT), 12-40
- Status Register, DMA (DMA\_STAT), 13-62
- Status Register, DMC (DMC\_STAT), 11-19
- Status Register, DPM (DPM\_STAT), 5-15
- Status Register, EPPI (EPPI\_STAT), 31-69
- Status Register, L2CTL (L2CTL\_STAT), 10-20
- Status Register, LP (LP\_STAT), 28-21
- Status Register, PWM (PWM\_STAT), 18-56
- Status Register, RCU (RCU\_STAT), 33-7, 34-69
- Status Register, RSI (RSI\_XFRSTAT), 24-54
- Status Register, SDU (SDU\_STAT), 35-16
- Status Register, SPI (SPI\_STAT), 25-52
- Status Register, SPU (SPU\_STAT), 4-9
- Status Register, UART (UART\_STAT), 19-34
- Status, PVP (PVP\_STAT), 30-117
- Stop Configuration Clear Register, TIMER (TIMER\_STOP\_CFG\_CLR), 15-33
- Stop Configuration Register, TIMER (TIMER\_STOP\_CFG), 15-32
- Stop Configuration Set Register, TIMER (TIMER\_STOP\_CFG\_SET), 15-32
- SVECT Lock Register, RCU (RCU\_SVECT\_LCK), 33-13
- SWU
  - block diagram, 36-5
- SWU0 Event (Masters), 8-6, 36-3
- SWU0 Event (Slaves), 8-11, 36-3
- SWU0 Event interrupt, 7-9, 36-2
- SWU1 Event (Masters), 8-6, 36-3
- SWU1 Event (Slaves), 8-11, 36-3
- SWU1 Event interrupt, 7-10, 36-2
- SWU2 Event (Masters), 8-6, 36-3
- SWU2 Event (Slaves), 8-11, 36-3
- SWU2 Event interrupt, 7-10, 36-2
- SWU3 Event (Masters), 8-6, 36-3
- SWU3 Event (Slaves), 8-11, 36-3
- SWU3 Event interrupt, 7-10, 36-2
- SWU4 Event (Masters), 8-6, 36-3
- SWU4 Event (Slaves), 8-11, 36-3
- SWU4 Event interrupt, 7-10, 36-2
- SWU5 Event (Masters), 8-6, 36-3
- SWU5 Event (Slaves), 8-11, 36-4
- SWU5 Event interrupt, 7-10, 36-3
- SWU6 Event (Masters), 8-6, 36-3
- SWU6 Event (Slaves), 8-11, 36-4
- SWU6 Event interrupt, 7-10, 36-3
- SWU\_CNTn (Count Register n, SWU), 36-23
- SWU\_CTLn (Control Register n, SWU), 36-16
- SWU\_CURn (Current Register n, SWU), 36-26
- SWU\_GCTL (Global Control Register, SWU), 36-10

SWU\_GSTAT (Global Status Register, SWU), 36-12  
 SWU\_HISTn (Bandwidth History Register n, SWU), 36-25  
 SWU\_IDn (ID Register n, SWU), 36-23  
 SWU\_LAn (Lower Address Register n, SWU), 36-21  
 SWU\_TARGn (Target Register n, SWU), 36-24  
 SWU\_UAn (Upper Address Register n, SWU), 36-22  
 Sync Pulse Width Register, PWM  
 (PWM\_SYNC\_WID), 18-68  
 System Clock Buffer Disable Register, DPM  
 (DPM\_SCBF\_DIS), 5-21  
 System Interface Disable Register, RCU  
 (RCU\_SIDIS), 33-11  
 System Interface Status Register, RCU  
 (RCU\_SISTAT), 33-12

## T

Target Register n, SWU (SWU\_TARGn), 36-24  
 TCNTL (Timer Control Register, TMR), 6-4  
 TCOUNT (Timer Count Register, TMR), 6-7  
 Temporary Mailbox Disable Register, CAN  
 (CAN\_MBTD), 21-67  
 Testmode Register, USB (USB\_TESTMODE), 22-111  
 THCn Clip Max Threshold, PVP  
 (PVP\_THCn\_CMAXTH), 30-195  
 THCn Clip Min Threshold, PVP  
 (PVP\_THCn\_CMINTH), 30-194  
 THCn Configuration, PVP (PVP\_THCn\_CFG), 30-187  
 THCn Control, PVP (PVP\_THCn\_CTL), 30-189  
 THCn Histogram Counter Value 0, PVP  
 (PVP\_THCn\_HCNT0\_STAT), 30-211  
 THCn Histogram Counter Value 1, PVP  
 (PVP\_THCn\_HCNT1\_STAT), 30-212  
 THCn Histogram Counter Value 10, PVP  
 (PVP\_THCn\_HCNT10\_STAT), 30-217  
 THCn Histogram Counter Value 11, PVP  
 (PVP\_THCn\_HCNT11\_STAT), 30-218  
 THCn Histogram Counter Value 12, PVP  
 (PVP\_THCn\_HCNT12\_STAT), 30-218  
 THCn Histogram Counter Value 13, PVP  
 (PVP\_THCn\_HCNT13\_STAT), 30-219  
 THCn Histogram Counter Value 14, PVP  
 (PVP\_THCn\_HCNT14\_STAT), 30-220  
 THCn Histogram Counter Value 15, PVP

(PVP\_THCn\_HCNT15\_STAT), 30-220  
 THCn Histogram Counter Value 2, PVP  
 (PVP\_THCn\_HCNT2\_STAT), 30-212  
 THCn Histogram Counter Value 3, PVP  
 (PVP\_THCn\_HCNT3\_STAT), 30-213  
 THCn Histogram Counter Value 4, PVP  
 (PVP\_THCn\_HCNT4\_STAT), 30-214  
 THCn Histogram Counter Value 5, PVP  
 (PVP\_THCn\_HCNT5\_STAT), 30-214  
 THCn Histogram Counter Value 6, PVP  
 (PVP\_THCn\_HCNT6\_STAT), 30-215  
 THCn Histogram Counter Value 7, PVP  
 (PVP\_THCn\_HCNT7\_STAT), 30-215  
 THCn Histogram Counter Value 8, PVP  
 (PVP\_THCn\_HCNT8\_STAT), 30-216  
 THCn Histogram Counter Value 9, PVP  
 (PVP\_THCn\_HCNT9\_STAT), 30-217  
 THCn Histogram Frame Count Status, PVP  
 (PVP\_THCn\_HFCNT\_STAT), 30-211  
 THCn Histogram Frame Count, PVP  
 (PVP\_THCn\_HFCNT), 30-192  
 THCn Histogram Horizontal Count, PVP  
 (PVP\_THCn\_HHCNT), 30-207  
 THCn Histogram Horizontal Position, PVP  
 (PVP\_THCn\_HHPOS), 30-205  
 THCn Histogram Vertical Count, PVP  
 (PVP\_THCn\_HVCNT), 30-207  
 THCn Histogram Vertical Position, PVP  
 (PVP\_THCn\_HVPOS), 30-206  
 THCn Max Clip Value, PVP  
 (PVP\_THCn\_CMAXVAL), 30-195  
 THCn Max RLE Reports, PVP  
 (PVP\_THCn\_RMAXREP), 30-193  
 THCn Min Clip Value, PVP  
 (PVP\_THCn\_CMINVAL), 30-193  
 THCn Number of RLE Reports, PVP  
 (PVP\_THCn\_RREP\_STAT), 30-221  
 THCn RLE Horizontal Count, PVP  
 (PVP\_THCn\_RHCNT), 30-209  
 THCn RLE Horizontal Position, PVP  
 (PVP\_THCn\_RHPOS), 30-208  
 THCn RLE Vertical Count, PVP  
 (PVP\_THCn\_RVCNT), 30-210  
 THCn RLE Vertical Position, PVP  
 (PVP\_THCn\_RVPOS), 30-209

- THCn Threshold Value 0, PVP (PVP\_THCn\_TH0), 30-196
- THCn Threshold Value 1, PVP (PVP\_THCn\_TH1), 30-196
- THCn Threshold Value 10, PVP (PVP\_THCn\_TH10), 30-202
- THCn Threshold Value 11, PVP (PVP\_THCn\_TH11), 30-202
- THCn Threshold Value 12, PVP (PVP\_THCn\_TH12), 30-203
- THCn Threshold Value 13, PVP (PVP\_THCn\_TH13), 30-204
- THCn Threshold Value 14, PVP (PVP\_THCn\_TH14), 30-204
- THCn Threshold Value 15, PVP (PVP\_THCn\_TH15), 30-205
- THCn Threshold Value 2, PVP (PVP\_THCn\_TH2), 30-197
- THCn Threshold Value 3, PVP (PVP\_THCn\_TH3), 30-198
- THCn Threshold Value 4, PVP (PVP\_THCn\_TH4), 30-198
- THCn Threshold Value 5, PVP (PVP\_THCn\_TH5), 30-199
- THCn Threshold Value 6, PVP (PVP\_THCn\_TH6), 30-199
- THCn Threshold Value 7, PVP (PVP\_THCn\_TH7), 30-200
- THCn Threshold Value 8, PVP (PVP\_THCn\_TH8), 30-201
- THCn Threshold Value 9, PVP (PVP\_THCn\_TH9), 30-201
- This register contains bit to dis CCS gen, RSI (RSI\_CEATA), 24-61
- Time Stamp Addend Register, EMAC (EMAC\_TM\_ADDEND), 23-212
- Time Stamp Auxiliary TM Seconds Register, EMAC (EMAC\_TM\_AUXSTMP\_SEC), 23-220
- Time Stamp Auxiliary TS Nano Seconds Register, EMAC (EMAC\_TM\_AUXSTMP\_NSEC), 23-220
- Time Stamp Control Register, EMAC (EMAC\_TM\_CTL), 23-203
- Time Stamp High Second Register, EMAC (EMAC\_TM\_HISEC), 23-214
- Time Stamp Low Seconds Register, EMAC (EMAC\_TM\_SEC), 23-209
- Time Stamp Nano Seconds Register, EMAC (EMAC\_TM\_NSEC), 23-209
- Time Stamp Nano Seconds Update Register, EMAC (EMAC\_TM\_NSECUPDT), 23-211
- Time Stamp PPS Interval Register, EMAC (EMAC\_TM\_PPSINTVL), 23-221
- Time Stamp Seconds Update Register, EMAC (EMAC\_TM\_SECUPDT), 23-210
- Time Stamp Status Register, EMAC (EMAC\_TM\_STMPSTAT), 23-215
- Time Stamp Sub Second Increment Register, EMAC (EMAC\_TM\_SUBSEC), 23-208
- Time Stamp Target Time Nano Seconds Register, EMAC (EMAC\_TM\_NTGMT), 23-213
- Time Stamp Target Time Seconds Register, EMAC (EMAC\_TM\_TGMT), 23-212
- Timer 0 Period Register, PWM (PWM\_TM0), 18-69
- Timer 1 Period Register, PWM (PWM\_TM1), 18-70
- Timer 2 Period Register, PWM (PWM\_TM2), 18-71
- Timer 3 Period Register, PWM (PWM\_TM3), 18-72
- Timer 4 Period Register, PWM (PWM\_TM4), 18-73
- Timer Control Register, TMR (TCNTL), 6-4
- Timer Count Register, TMR (TCOUNT), 6-7
- Timer n Configuration Register, TIMER (TIMER\_TMRn\_CFG), 15-43
- Timer n Counter Register, TIMER (TIMER\_TMRn\_CNT), 15-48
- Timer n Delay Register, TIMER (TIMER\_TMRn\_DLY), 15-49
- Timer n Period Register, TIMER (TIMER\_TMRn\_PER), 15-48
- Timer n Width Register, TIMER (TIMER\_TMRn\_WID), 15-49
- Timer Period Register, TMR (TPERIOD), 6-5
- Timer Scale Register, TMR (TSCALE), 6-6
- TIMER0 Status interrupt, 7-4, 15-3
- TIMER0 Timer 0 (Masters), 8-2, 15-3
- TIMER0 Timer 0 (Slaves), 8-7, 15-4
- TIMER0 Timer 0 interrupt, 7-3, 15-3
- TIMER0 Timer 1 (Masters), 8-2, 15-4
- TIMER0 Timer 1 (Slaves), 8-7, 15-4
- TIMER0 Timer 1 interrupt, 7-3, 15-3
- TIMER0 Timer 2 (Masters), 8-2, 15-4
- TIMER0 Timer 2 (Slaves), 8-7, 15-4

- TIMER0 Timer 2 interrupt, 7-3, 15-3
- TIMER0 Timer 3 (Masters), 8-2, 15-4
- TIMER0 Timer 3 (Slaves), 8-7, 15-4
- TIMER0 Timer 3 interrupt, 7-3, 15-3
- TIMER0 Timer 4 (Masters), 8-2, 15-4
- TIMER0 Timer 4 (Slaves), 8-7, 15-4
- TIMER0 Timer 4 interrupt, 7-4, 15-3
- TIMER0 Timer 5 (Masters), 8-2, 15-4
- TIMER0 Timer 5 (Slaves), 8-7, 15-4
- TIMER0 Timer 5 interrupt, 7-4, 15-3
- TIMER0 Timer 6 (Masters), 8-2, 15-4
- TIMER0 Timer 6 (Slaves), 8-7, 15-4
- TIMER0 Timer 6 interrupt, 7-4, 15-3
- TIMER0 Timer 7 (Masters), 8-2, 15-4
- TIMER0 Timer 7 (Slaves), 8-7, 15-4
- TIMER0 Timer 7 interrupt, 7-4, 15-3
- TIMER\_BCAST\_DLY (Broadcast Delay Register, TIMER), 15-42
- TIMER\_BCAST\_PER (Broadcast Period Register, TIMER), 15-40
- TIMER\_BCAST\_WID (Broadcast Width Register, TIMER), 15-41
- TIMER\_DATA\_ILAT (Data Interrupt Latch Register, TIMER), 15-36
- TIMER\_DATA\_IMSK (Data Interrupt Mask Register, TIMER), 15-34
- TIMER\_ERR\_TYPE (Error Type Status Register, TIMER), 15-38
- TIMER\_REVID (Revision ID Register, TIMER), 15-29
- TIMER\_RUN (Run Register, TIMER), 15-30
- TIMER\_RUN\_CLR (Run Clear Register, TIMER), 15-31
- TIMER\_RUN\_SET (Run Set Register, TIMER), 15-30
- TIMER\_STAT\_ILAT (Status Interrupt Latch Register, TIMER), 15-37
- TIMER\_STAT\_IMSK (Status Interrupt Mask Register, TIMER), 15-34
- TIMER\_STOP\_CFG (Stop Configuration Register, TIMER), 15-32
- TIMER\_STOP\_CFG\_CLR (Stop Configuration Clear Register, TIMER), 15-33
- TIMER\_STOP\_CFG\_SET (Stop Configuration Set Register, TIMER), 15-32
- TIMER\_TMRn\_CFG (Timer n Configuration Register, TIMER), 15-43
- TIMER\_TMRn\_CNT (Timer n Counter Register, TIMER), 15-48
- TIMER\_TMRn\_DLY (Timer n Delay Register, TIMER), 15-49
- TIMER\_TMRn\_PER (Timer n Period Register, TIMER), 15-48
- TIMER\_TMRn\_WID (Timer n Width Register, TIMER), 15-49
- TIMER\_TRG\_IE (Trigger Slave Enable Register, TIMER), 15-36
- TIMER\_TRG\_MSK (Trigger Master Mask Register, TIMER), 15-35
- Timing 0 Register, DMC (DMC\_TR0), 11-29
- Timing 1 Register, DMC (DMC\_TR1), 11-31
- Timing 2 Register, DMC (DMC\_TR2), 11-32
- Timing Register, CAN (CAN\_TIMING), 21-47
- TPERIOD (Timer Period Register, TMR), 6-5
- transfer size (TxferSize);TxferSize (transfer size), 22-28, 22-30
- Transmission Acknowledge 1 Register, CAN (CAN\_TA1), 21-34
- Transmission Acknowledge 2 Register, CAN (CAN\_TA2), 21-41
- Transmission Request Reset 1 Register, CAN (CAN\_TRR1), 21-34
- Transmission Request Reset 2 Register, CAN (CAN\_TRR2), 21-41
- Transmission Request Set 1 Register, CAN (CAN\_TRS1), 21-33
- Transmission Request Set 2 Register, CAN (CAN\_TRS2), 21-40
- Transmit Address/Insert Pulse Register, UART (UART\_TAIP), 19-51
- Transmit Buffer, LP (LP\_TX), 28-24
- Transmit Control Register, SPI (SPI\_TXCTL), 25-34
- Transmit Counter Register, UART (UART\_TXCNT), 19-53
- Transmit FIFO Address Register, USB (USB\_TXFIFOADDR), 22-121
- Transmit FIFO Data Register, SPI (SPI\_TFIFO), 25-61
- Transmit FIFO Size Register, USB (USB\_TXFIFOSZ), 22-117
- Transmit Hold Register, UART (UART\_THR), 19-50

- Transmit Interrupt Enable Register, USB (USB\_INTRTXE), 22-99
- Transmit Interrupt Register, USB (USB\_INTRTX), 22-93
- Transmit Shift Register, UART (UART\_TSR), 19-51
- Transmitted Word Count Register, SPI (SPI\_TWC), 25-45
- Transmitted Word Count Reload Register, SPI (SPI\_TWCR), 25-46
- Transparency Color Register, PIXC (PIXC\_TC), 32-50
- Trigger Master Mask Register, TIMER (TIMER\_TRG\_MSK), 15-35
- Trigger Slave Enable Register, TIMER (TIMER\_TRG\_IE), 15-36
- Trip Config Register, PWM (PWM\_TRIPCFG), 18-51
- TRU0 Interrupt 0 interrupt, 7-9
- TRU0 Interrupt 1 interrupt, 7-9
- TRU0 Interrupt 2 interrupt, 7-9
- TRU0 Interrupt 3 interrupt, 7-9
- TRU0 Interrupt Request 0 (Slaves), 8-7
- TRU0 Interrupt Request 1 (Slaves), 8-7
- TRU0 Interrupt Request 2 (Slaves), 8-7
- TRU0 Interrupt Request 3 (Slaves), 8-7
- TRU\_ERRADDR (Error Address Register, TRU), 8-16
- TRU\_GCTL (Global Control Register, TRU), 8-18
- TRU\_MTR (Master Trigger Register, TRU), 8-15
- TRU\_REVID (Revision ID Register, TRU), 8-18
- TRU\_SSRn (Slave Select Register, TRU), 8-14
- TRU\_STAT (Status Information Register, TRU), 8-17
- TSCALE (Timer Scale Register, TMR), 6-6
- TWI0 Data Interrupt interrupt, 7-4, 20-3
- TWI1 Data Interrupt interrupt, 7-4, 20-3
- TWI\_CLKDIV (SCL Clock Divider Register, TWI), 20-19
- TWI\_CTL (Control Register, TWI), 20-19
- TWI\_FIFOCTL (FIFO Control Register, TWI), 20-37
- TWI\_FIFOSTAT (FIFO Status Register, TWI), 20-38
- TWI\_IMSK (Interrupt Mask Register, TWI), 20-35
- TWI\_ISTAT (Interrupt Status Register, TWI), 20-31
- TWI\_MSTRADDR (Master Mode Address Register, TWI), 20-31
- TWI\_MSTRCTL (Master Mode Control Registers, TWI), 20-24
- TWI\_MSTRSTAT (Master Mode Status Register, TWI), 20-27
- TWI\_RXDATA16 (Rx Data Double-Byte Register, TWI), 20-42
- TWI\_RXDATA8 (Rx Data Single-Byte Register, TWI), 20-41
- TWI\_SLVADDR (Slave Mode Address Register, TWI), 20-23
- TWI\_SLVCTL (Slave Mode Control Register, TWI), 20-21
- TWI\_SLVSTAT (Slave Mode Status Register, TWI), 20-22
- TWI\_TXDATA16 (Tx Data Double-Byte Register, TWI), 20-40
- TWI\_TXDATA8 (Tx Data Single-Byte Register, TWI), 20-40
- Tx 1024- to Max-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX1024TOMAX\_GB), 23-152
- Tx 128- to 255-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX128TO255\_GB), 23-150
- Tx 256- to 511-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX256TO511\_GB), 23-150
- Tx 512- to 1023-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX512TO1023\_GB), 23-151
- Tx 64-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX64\_GB), 23-149
- Tx 65- to 127-Byte Frames (Good/Bad) Register, EMAC (EMAC\_TX65TO127\_GB), 23-149
- Tx Broadcast Frames (Good) Register, EMAC (EMAC\_TXBCASTFRM\_G), 23-147
- Tx Broadcast Frames (Good/Bad) Register, EMAC (EMAC\_TXBCASTFRM\_GB), 23-153
- Tx Carrier Error Register, EMAC (EMAC\_TXCARR\_ERR), 23-158
- Tx Data Double-Byte Register, TWI (TWI\_TXDATA16), 20-40
- Tx Data Single-Byte Register, TWI (TWI\_TXDATA8), 20-40
- Tx Deferred Register, EMAC (EMAC\_TXDEFERRED), 23-156
- Tx Excess Collision Register, EMAC (EMAC\_TXEXCESSCOL), 23-157
- Tx Excess Deferral Register, EMAC (EMAC\_TXEXCESSDEF), 23-159
- Tx Frame Count (Good) Register, EMAC

(EMAC\_TXFRMCNT\_G), 23-159  
 Tx Frame Count (Good/Bad) Register, EMAC  
 (EMAC\_TXFRMCNT\_GB), 23-147  
 Tx Late Collision Register, EMAC  
 (EMAC\_TXLATECOL), 23-156  
 Tx Multicast Frames (Good) Register, EMAC  
 (EMAC\_TXMCASTFRM\_G), 23-148  
 Tx Multicast Frames (Good/Bad) Register, EMAC  
 (EMAC\_TXMCASTFRM\_GB), 23-153  
 Tx Multiple Collision (Good) Register, EMAC  
 (EMAC\_TXMULTCOL\_G), 23-155  
 Tx OCT Count (Good/Bad) Register, EMAC  
 (EMAC\_TXOCTCNT\_GB), 23-146  
 Tx Octet Count (Good) Register, EMAC  
 (EMAC\_TXOCTCNT\_G), 23-158  
 Tx Pause Frame Register, EMAC  
 (EMAC\_TXPAUSEFRM), 23-160  
 Tx Single Collision (Good) Register, EMAC  
 (EMAC\_TXSNGCOL\_G), 23-155  
 Tx Underflow Error Register, EMAC  
 (EMAC\_TXUNDR\_ERR), 23-154  
 Tx Unicast Frames (Good/Bad) Register, EMAC  
 (EMAC\_TXUCASTFRM\_GB), 23-152  
 Tx VLAN Frames (Good) Register, EMAC  
 (EMAC\_TXVLANFRM\_G), 23-161

## U

UART0 Receive DMA (Masters), 8-4, 19-4  
 UART0 Receive DMA (Slaves), 8-8, 19-4  
 UART0 Receive DMA interrupt, 7-7, 19-4  
 UART0 Status interrupt, 7-7, 19-4  
 UART0 Transmit DMA (Masters), 8-4, 19-4  
 UART0 Transmit DMA (Slaves), 8-8, 19-4  
 UART0 Transmit DMA interrupt, 7-6, 19-4  
 UART1 Receive DMA (Masters), 8-4, 19-4  
 UART1 Receive DMA (Slaves), 8-8, 19-5  
 UART1 Receive DMA interrupt, 7-7, 19-4  
 UART1 Status interrupt, 7-7, 19-4  
 UART1 Transmit DMA (Masters), 8-4, 19-4  
 UART1 Transmit DMA (Slaves), 8-8, 19-4  
 UART1 Transmit DMA interrupt, 7-7, 19-4  
 UART\_CLK (Clock Rate Register, UART), 19-40  
 UART\_CTL (Control Register, UART), 19-26  
 UART\_IMSK (Interrupt Mask Register, UART),  
 19-41

UART\_IMSK\_CLR (Interrupt Mask Clear Register,  
 UART), 19-47  
 UART\_IMSK\_SET (Interrupt Mask Set Register,  
 UART), 19-45  
 UART\_RBR (Receive Buffer Register, UART), 19-49  
 UART\_REVID (Revision ID Register, UART), 19-25  
 UART\_RSR (Receive Shift Register, UART), 19-52  
 UART\_RXCNT (Receive Counter Register, UART),  
 19-53  
 UART\_SCR (Scratch Register, UART), 19-39  
 UART\_STAT (Status Register, UART), 19-34  
 UART\_TAIP (Transmit Address/Insert Pulse Register,  
 UART), 19-51  
 UART\_THR (Transmit Hold Register, UART), 19-50  
 UART\_TSR (Transmit Shift Register, UART), 19-51  
 UART\_TXCNT (Transmit Counter Register, UART),  
 19-53  
 UDS Configuration, PVP (PVP\_UDS\_CFG), 30-160  
 UDS Control, PVP (PVP\_UDS\_CTL), 30-161  
 UDS HAVG, PVP (PVP\_UDS\_HAVG), 30-163  
 UDS Output HCNT, PVP (PVP\_UDS\_OHCNT),  
 30-162  
 UDS Output VCNT, PVP (PVP\_UDS\_OVCNT),  
 30-162  
 UDS VAVG, PVP (PVP\_UDS\_VAVG), 30-164  
 Universal Counter Configuration Mode Register, CAN  
 (CAN\_UCCNF), 21-73  
 Universal Counter Register, CAN (CAN\_UCCNT),  
 21-72  
 Universal Counter Reload/Capture Register, CAN  
 (CAN\_UCRC), 21-72  
 Upper Address Register n, SWU (SWU\_UAn), 36-22  
 USB0 DMA Status/Transfer Complete (Masters), 8-6,  
 22-9  
 USB0 DMA Status/Transfer Complete interrupt, 7-9,  
 22-9  
 USB0 Status/FIFO Data Ready interrupt, 7-9, 22-8  
 USB\_BAT\_CHG (Battery Charging Control Register,  
 USB), 22-193  
 USB\_CT\_HHSRTN (Host High Speed Return to Nor-  
 mal Register, USB), 22-181  
 USB\_CT\_HSBT (High Speed Timeout Register,  
 USB), 22-182  
 USB\_CT\_UCH (Chirp Timeout Register, USB),  
 22-181

- USB\_DEV\_CTL (Device Control Register, USB), 22-114
- USB\_DMA\_IRQ (DMA Interrupt Register, USB), 22-173
- USB\_DMAn\_ADDR (DMA Channel n Address Register, USB), 22-179
- USB\_DMAn\_CNT (DMA Channel n Count Register, USB), 22-179
- USB\_DMAn\_CTL (DMA Channel n Control Register, USB), 22-176
- USB\_EP0\_CFGDATAn (EP0 Configuration Information Register, USB), 22-171
- USB\_EP0\_CNTn (EP0 Number of Received Bytes Register, USB), 22-161
- USB\_EP0\_CSRn\_H (EP0 Configuration and Status (Host) Register, USB), 22-133
- USB\_EP0\_CSRn\_P (EP0 Configuration and Status (Peripheral) Register, USB), 22-137
- USB\_EP0\_NAKLIMITn (EP0 NAK Limit Register, USB), 22-165
- USB\_EP0\_TYPEn (EP0 Connection Type Register, USB), 22-165
- USB\_EPINFO (Endpoint Information Register, USB), 22-122
- USB\_EPn\_RXCNT (EPn Number of Bytes Received Register, USB), 22-161
- USB\_EPn\_RXCSR\_H (EPn Receive Configuration and Status (Host) Register, USB), 22-150
- USB\_EPn\_RXCSR\_P (EPn Receive Configuration and Status (Peripheral) Register, USB), 22-156
- USB\_EPn\_RXINTERVAL (EPn Receive Polling Interval Register, USB), 22-170
- USB\_EPn\_RXMAXP (EPn Receive Maximum Packet Length Register, USB), 22-149
- USB\_EPn\_RXTYPE (EPn Receive Type Register, USB), 22-167
- USB\_EPn\_TXCSR\_H (EPn Transmit Configuration and Status (Host) Register, USB), 22-140
- USB\_EPn\_TXCSR\_P (EPn Transmit Configuration and Status (Peripheral) Register, USB), 22-145
- USB\_EPn\_TXINTERVAL (EPn Transmit Polling Interval Register, USB), 22-166
- USB\_EPn\_TXMAXP (EPn Transmit Maximum Packet Length Register, USB), 22-132
- USB\_EPn\_TXTYPE (EPn Transmit Type Register, USB), 22-162
- USB\_FADDR (Function Address Register, USB), 22-89
- USB\_FIFOBn (FIFO Byte (8-Bit) Register, USB), 22-112
- USB\_FIFOHn (FIFO Half-Word (16-Bit) Register, USB), 22-113
- USB\_FIFOn (FIFO Word (32-Bit) Register, USB), 22-114
- USB\_FRAME (Frame Number Register, USB), 22-109
- USB\_FS\_EOF1 (Full-Speed EOF 1 Register, USB), 22-125
- USB\_HS\_EOF1 (High-Speed EOF 1 Register, USB), 22-125
- USB\_IEN (Common Interrupts Enable Register, USB), 22-107
- USB\_INDEX (Index Register, USB), 22-110
- USB\_INTRRX (Receive Interrupt Register, USB), 22-96
- USB\_INTRRXE (Receive Interrupt Enable Register, USB), 22-102
- USB\_INTRTX (Transmit Interrupt Register, USB), 22-93
- USB\_INTRTXE (Transmit Interrupt Enable Register, USB), 22-99
- USB\_IRQ (Common Interrupts Register, USB), 22-105
- USB\_LINKINFO (Link Information Register, USB), 22-123
- USB\_LPM\_ATTR (LPM Attribute Register, USB), 22-183
- USB\_LPM\_CTL (LPM Control Register, USB), 22-184
- USB\_LPM\_FADDR (LPM Function Address Register, USB), 22-191
- USB\_LPM\_IEN (LPM Interrupt Enable Register, USB), 22-187
- USB\_LPM\_IRQ (LPM Interrupt Status Register, USB), 22-188
- USB\_LS\_EOF1 (Low-Speed EOF 1 Register, USB), 22-126
- USB\_MPn\_RXFUNCADDR (MPn Receive Function Address Register, USB), 22-130
- USB\_MPn\_RXHUBADDR (MPn Receive Hub Ad-

dress Register, USB), 22-130  
 USB\_MPn\_RXHUBPORT (MPn Receive Hub Port Register, USB), 22-131  
 USB\_MPn\_TXFUNCADDR (MPn Transmit Function Address Register, USB), 22-128  
 USB\_MPn\_TXHUBADDR (MPn Transmit Hub Address Register, USB), 22-128  
 USB\_MPn\_TXHUBPORT (MPn Transmit Hub Port Register, USB), 22-129  
 USB\_PHY\_CTL (PHY Control Register, USB), 22-194  
 USB\_PLL\_OSC (PLL and Oscillator Control Register, USB), 22-195  
 USB\_POWER (Power and Device Control Register, USB), 22-89  
 USB\_RAMINFO (RAM Information Register, USB), 22-123  
 USB\_RQPKTCNTn (EPn Request Packet Count Register, USB), 22-180  
 USB\_RXFIFOADDR (Receive FIFO Address Register, USB), 22-121  
 USB\_RXFIFOSZ (Receive FIFO Size Register, USB), 22-119  
 USB\_SOFT\_RST (Software Reset Register, USB), 22-127  
 USB\_TESTMODE (Testmode Register, USB), 22-111  
 USB\_TXFIFOADDR (Transmit FIFO Address Register, USB), 22-121  
 USB\_TXFIFOSZ (Transmit FIFO Size Register, USB), 22-117  
 USB\_VBUS\_CTL (VBUS Control Register, USB), 22-192  
 USB\_VPLEN (VBUS Pulse Length Register, USB), 22-124

## V

VBUS Control Register, USB (USB\_VBUS\_CTL), 22-192  
 VBUS Pulse Length Register, USB (USB\_VPLEN), 22-124  
 Vertical Delay Count Register, EPPI (EPPI\_VDLY), 31-74  
 Vertical Transfer Count Register, EPPI (EPPI\_VCNT), 31-73  
 VID\_CONN (Video Subsystem Connect Register,

VID), 29-8  
 Video Subsystem Connect Register, VID (VID\_CONN), 29-8  
 VLAN Tag Register, EMAC (EMAC\_VLANTAG), 23-122

## W

Wakeup Enable Register, DPM (DPM\_WAKE\_EN), 5-22  
 Wakeup Polarity Register, DPM (DPM\_WAKE\_POL), 5-23  
 Wakeup Status Register, DPM (DPM\_WAKE\_STAT), 5-24  
 Watchdog Timer Status Register, WDOG (WDOG\_STAT), 16-6  
 WDOG0 Expiration interrupt, 7-3, 16-2  
 WDOG1 Expiration interrupt, 7-3, 16-2  
 WDOG\_CNT (Count Register, WDOG), 16-5  
 WDOG\_CTL (Control Register, WDOG), 16-4  
 WDOG\_STAT (Watchdog Timer Status Register, WDOG), 16-6  
 Write Priority Count Register, L2CTL (L2CTL\_WPCR), 10-24  
 Write Protect Register n, SPU (SPU\_WPn), 4-10









